FASTMATH

Putting it all together: One perspective

Cameron W. Smith and Mark S. Shephard

Rensselaer Polytechnic Institute

Argonne NATIONAL LABORATORY

BERKELEY LAB

OAK RIDGE National Laboratory

Sandia National Laboratories

Massachusetts Institute of Technology

Rensselaer

SMU

USC University of Southern California

Putting it all together: One perspective

Why “One perspective”? Because you have to first:

- Define what functionality you provide
- Determine what supporting components are needed
- Understand the constraints that you must work within

Functionality to be provided

- Integrated, parallel, unstructured mesh adaptation

Tools to be ‘put together’

- FASTMath unstructured meshing tools and analysis codes, solvers, UQ tools, optimization tools, in-situ vis tools, etc.
- Application analysis components w/ required physics and math models

FASTMATH

Putting it all together: Constraints

The constraints

- The applications people must advance their application
- If they have an existing analysis tools, they expect to build on it
  - It may not be an optimal code for the long term goal
  - But, the large investment to date can not just be scrapped

Approach taken when there is an existing analysis code:

- Provide mechanisms to integrate needed tools with their code with little or no modification
- Work directly with application to gain understanding and trust
- Incrementally evolve tools on both sides to be more effective – always with direct consideration of advancing the capabilities of the application

FASTMATH

Putting it all together: Things Change

When requirements or constraints change the ‘best’ approach to ‘put it together’ changes

If the analysis code is found to be deficient we could use an existing infrastructure like MFEM or Albany… but, you must:

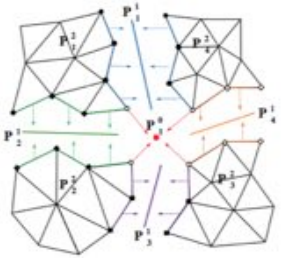
- Overcome all deficiencies
- Maintain backward compatibility
- Maintain or improve performance
- Demonstrate useful functionality quickly
- Support the ‘real’ physics – often harder than what mini-apps do

With that lets talk about providing application code developers with the ability to generate, solve and adapt unstructured meshes

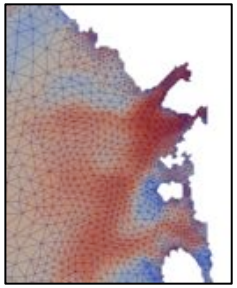
FASTMATH

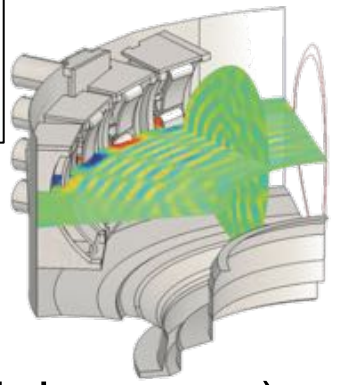
FASTMath tools in Unstructured Mesh Applications

- Parallel Unstructured Mesh Infrastructure (PUMI/MeshAdapt)
  - Operate on evolving meshes of any size in parallel
  - Geometry consistent, anisotropic, conforming adaptation

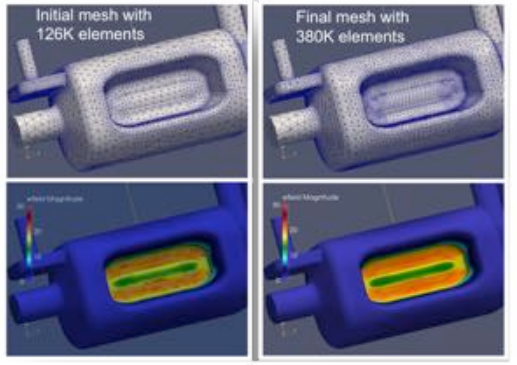


- Dynamic load balancing
  - Zoltan - graph and geometric load balancing
  - EnGPar – multicriteria partition improvement





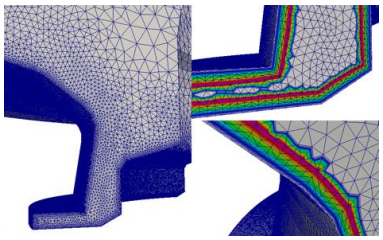
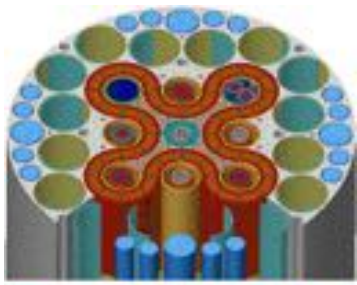
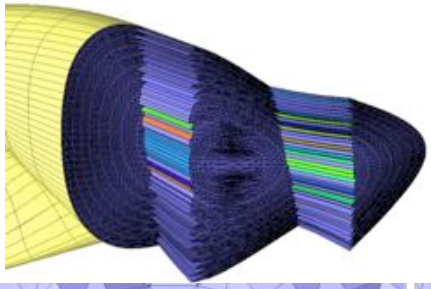
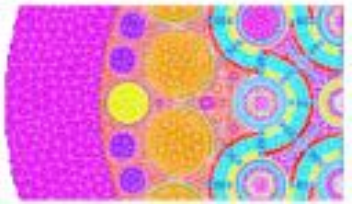
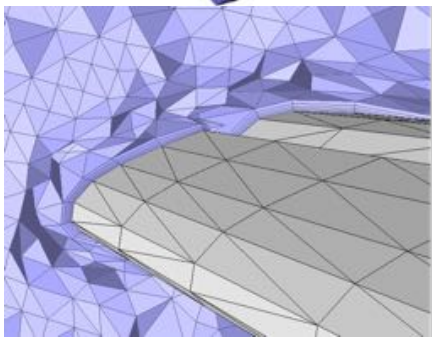
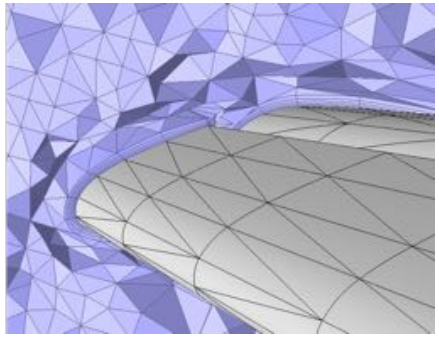
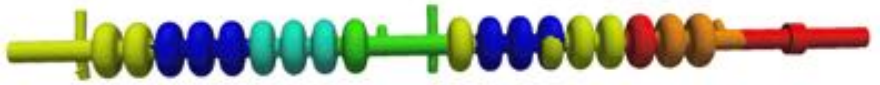
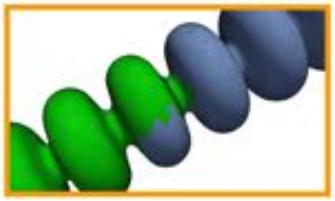
- Unstructured mesh analysis codes
  - MFEM, Albany, Phasta – introduced this morning
- Matrix systems “solvers” and time integrators
  - hypre, SuperLU, PETSc – introduced today (FASTMath has more)
- Optimization (TAO) – introduced today
- UQ (DAKOTA, UQTk)

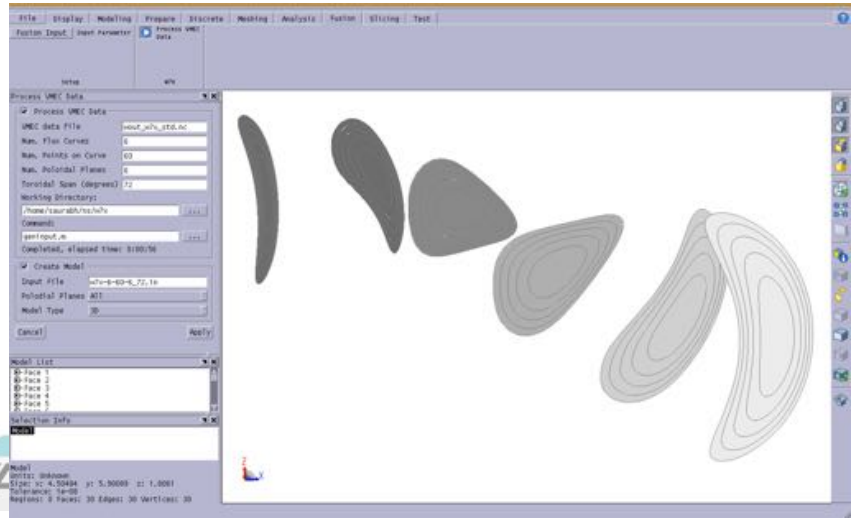


FASTMATH

Need to Integrate with Geometry and Meshing Tools

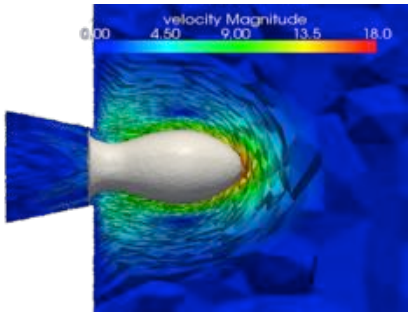
- Link with existing tools
  - Open source (e.g. Gmsh) and commercial
- Geometry and meshing components from Simmetrix
  - Direct links to CAD, geometry simplification and combination
  - Fully automatic parallel meshing
  - Mesh-based field manipulation
  - Customizable user interface

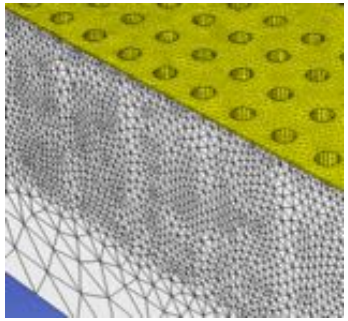









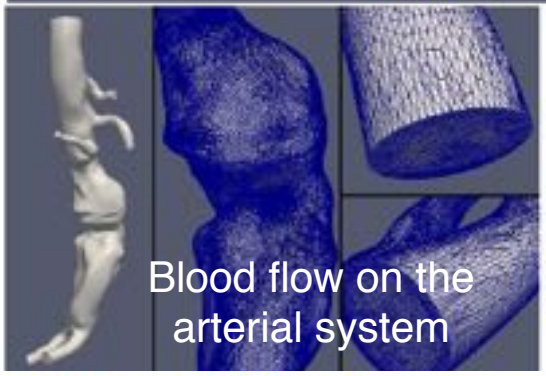


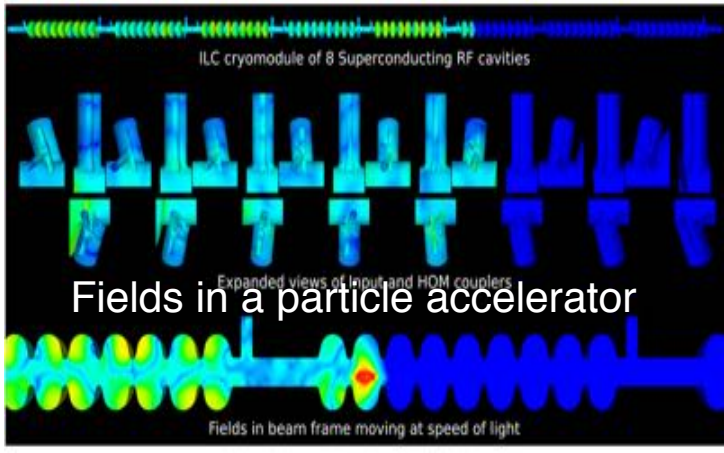
Parallel Adaptive Simulation Workflows

- Automation and adaptive methods critical to reliable simulations
- Approaches include providing
  - Meshing capabilities
  - File based integration – often done first
  - In-memory integration – when they see files are a bottleneck
  - Replace infrastructure but retain their numerics
  - Directly build in one of the FASTMath simulation infrastructures
    - MFEM and Albany designed for doing such developments







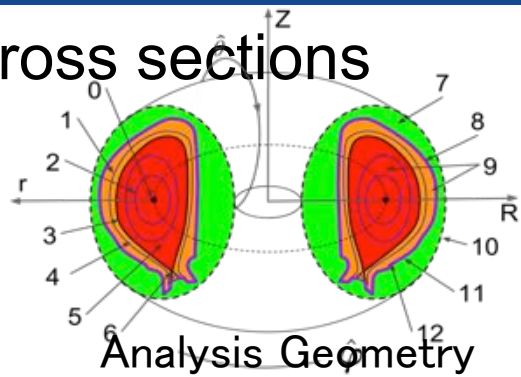


FASTMATH

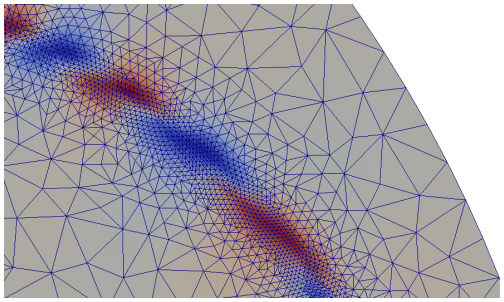
Providing Mesh Capabilities for Tokamak Plasma Simulations

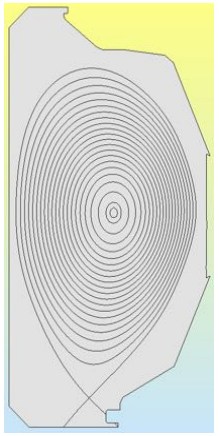
Geometry/meshing tool developed for tokamak cross sections

- "Field following" XGC meshes
- 2D to 3D and mesh adaptivity for M3D-C1
- Tokamak cross-section geometry
  - Analytic, spline or discrete physical components – ITER, DIII-D, Alcator C-MOD, NSTX, KSTAR,etc.
  - EFIT defined physics geometry
- Graphical user interface

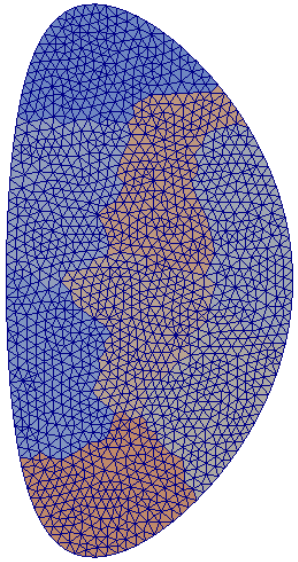


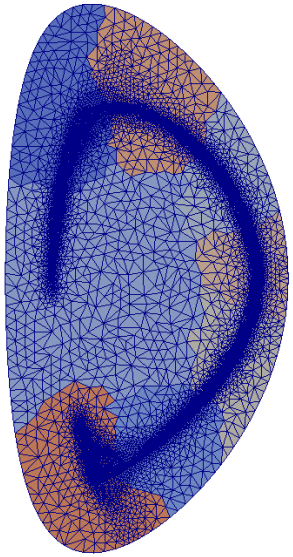
Analysis Geometry

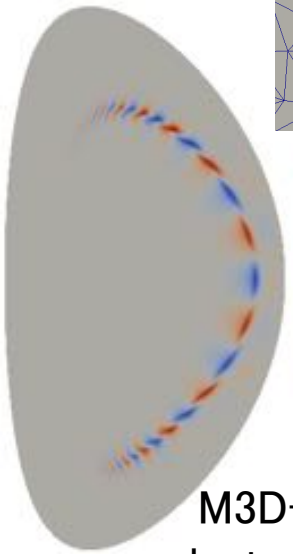


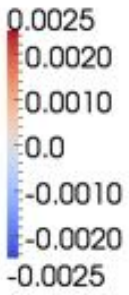


Field following XGC mesh

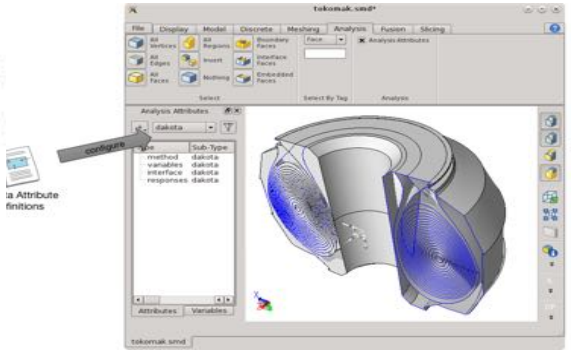








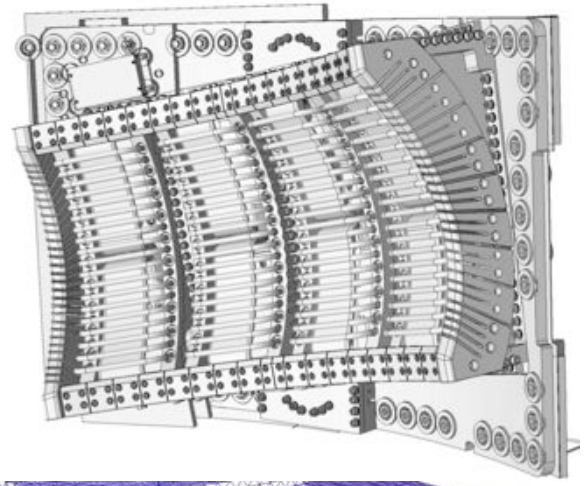
M3D-C1
adapted mesh

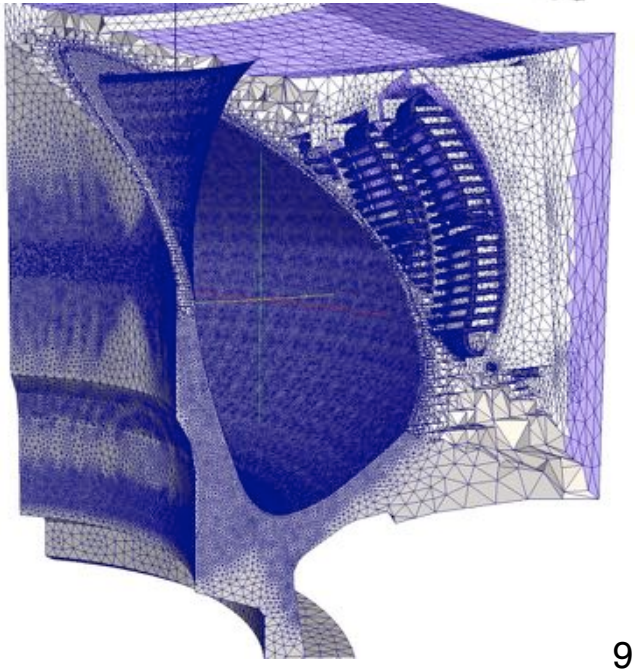


GUI

Providing Geometry, Meshing and Analysis for RF simulations

- Accurate RF simulations require
  - Complex Geometry
    - Detailed antenna CAD geometry
    - Extracted physics curves from EFIT
    - Faceted surfaces from coupled meshes
    - Analysis geometry combining CAD, physics geometry and faceted interfaces
  - Fully automatic meshing
  - Massively Parallel High-Order FEM
    - MFEM has been extended to address basic RF physics – hyper is core solver
    - Additional physics being developed
  - Integration into RF workflow
    - Coupling to RF workflow environment





FASTMATH

In-Memory Parallel-Adaptive Simulation Workflows

Goal: Develop, implement, and apply methods for massively parallel in-memory iterative execution of simulation components

In-memory Workflows

- MFEM
- M3D-C1
- SLAC ACE3P
- Albany
- PHASTA
- FUN3D (NASA)
- Proteus (DoD)

Most codes employ FASTMath matrix solution or load balancing components

Many have UQ extensions using Dakota

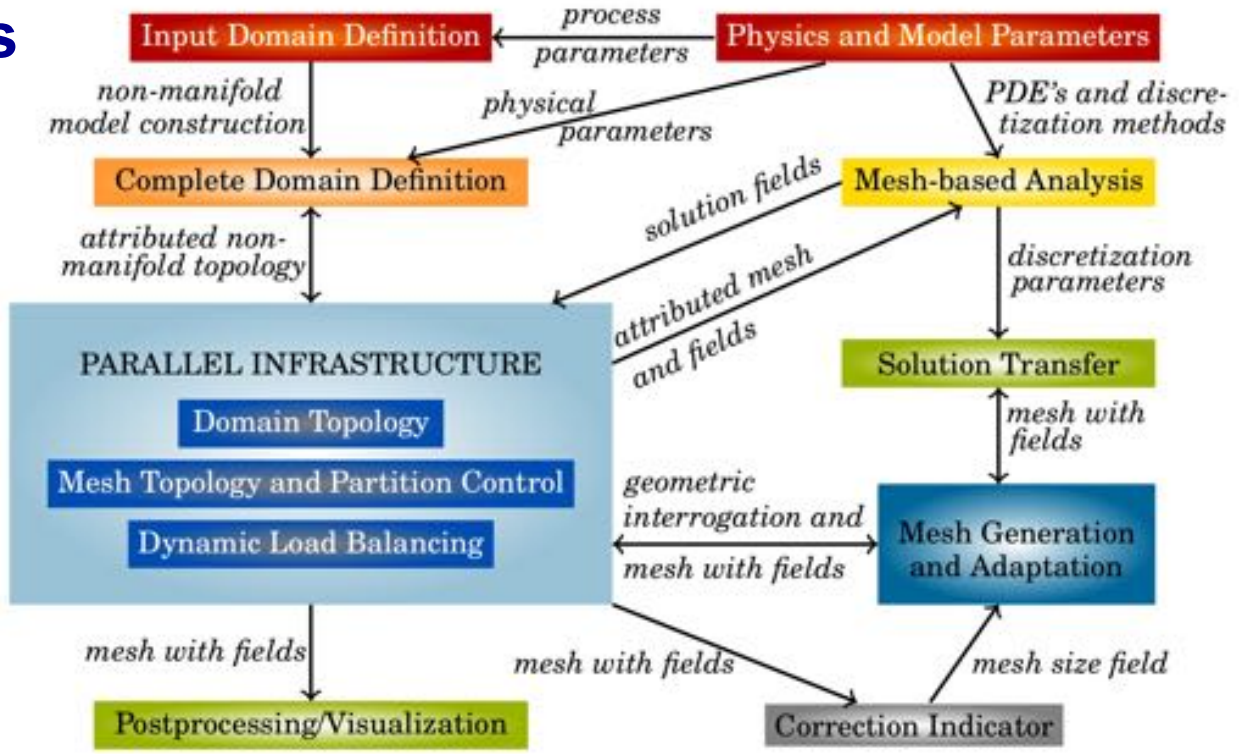


Fig. 1. Simulation infrastructure

Coordinating Component Interactions

Component Coupling: Component functional interfaces for control and field information passing and transformation

- Coordinate workflow with driver (e.g., `main()`) that calls major components
- Change/Add components with minimal development costs
- Abstract component complexities and implementation details
  - Keep information at the right abstraction level (i.e., not just the mesh!)
- Coupling across languages; vast majority are FORTRAN and C/C++

Challenges

- Elimination of file I/O for component coupling
- Managing Memory
- Selecting coupling method – depends on the component implementation!
- Balancing the work distribution
- Knowing when a rewrite is needed – when there is funding and skill!

FASTMATH

Eliminating File I/O

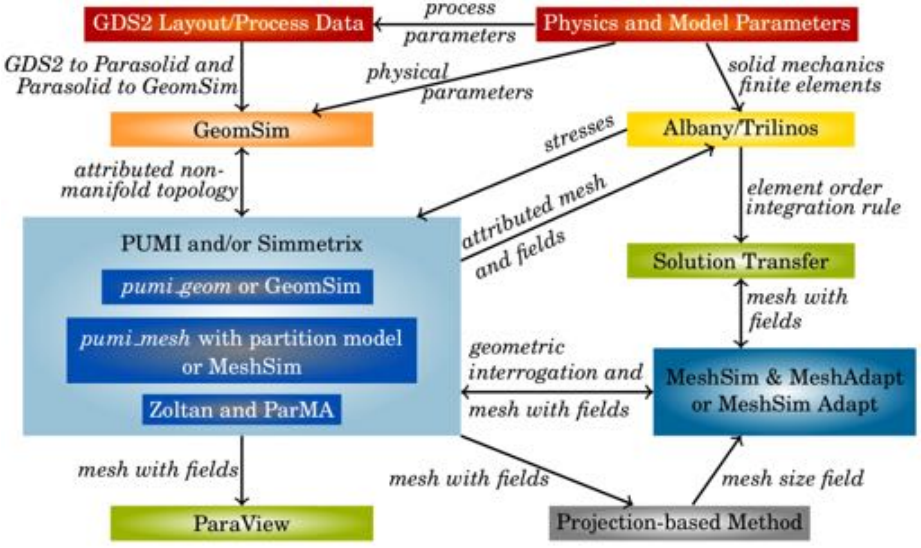
On massively parallel systems I/O dominates power consumption

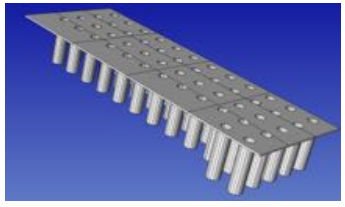
Use APIs and data-streams to keep inter-component information transfers in on-process memory

- Bulk Transfers
  - Transfer large sets of data using files or APIs
  - Ideal for components using POSIX files – minimal change for streams
  - Key Challenge - interface definition and memory management
- Atomic Transfers
  - Transfer a small set of data over a subset of the domain using APIs
  - Ideal for components with abstracted mesh and field interfaces
  - Key Challenge - understanding implementation++
- Component implementation determines the coupling approach

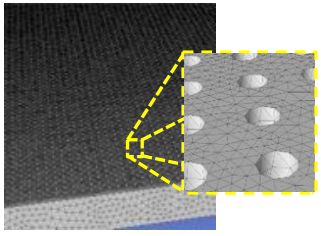
APIs or Data Streams?

Albany
templated C++ → bulk APIs



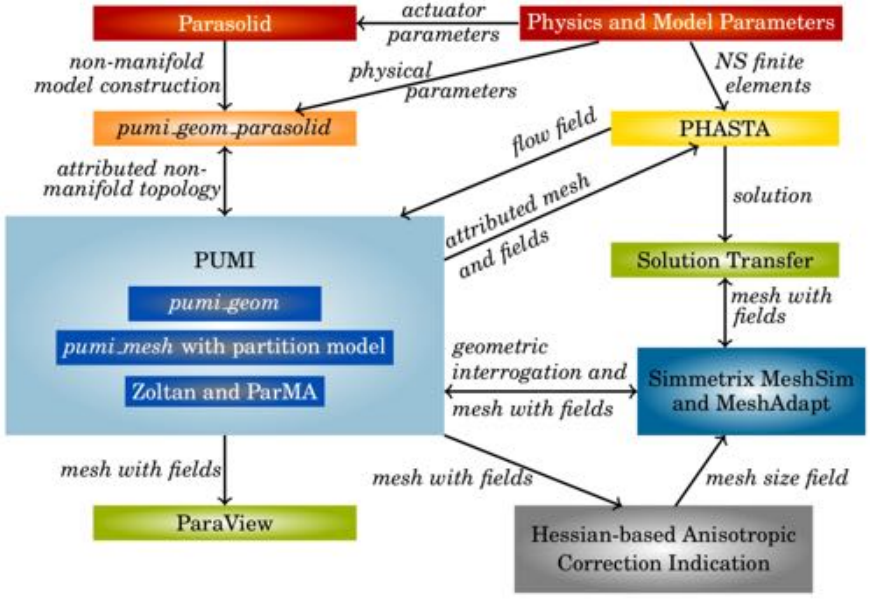


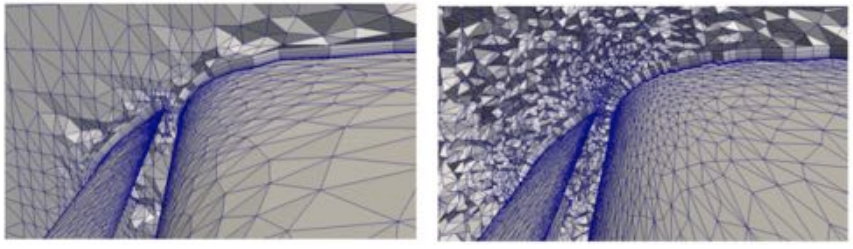
Model of liner film only.



1.1 billion element parallel mesh generated on BG/Q.

PHASTA
F77/90 + existing file I/O → data streams





Cut views of initial (leff) and adapted anistropic boundary layer meshes for NASA trap wing [Chitale et al. 2014]

FASTMATH

Balancing the work distribution in Adaptive Workflows

At >16Ki ranks, existing tools providing multi-level graph methods consume too much memory and fail; geometric methods have high cuts and are inefficient for analysis.

An approach that combines existing methods with **ParMA** diffusive improvement accounts for multiple criteria:

- Accounts for DOF on any mesh entity
- Analysis and partitioning is quicker

Goal of current **EnGPar** developments is to generalize methods

- Take advantage of graph methods and new hardware
- Broaden the areas of application to new applications (mesh based and others)

Partitioning to 1M Parts

Multiple tools needed to maintain partition quality at scale

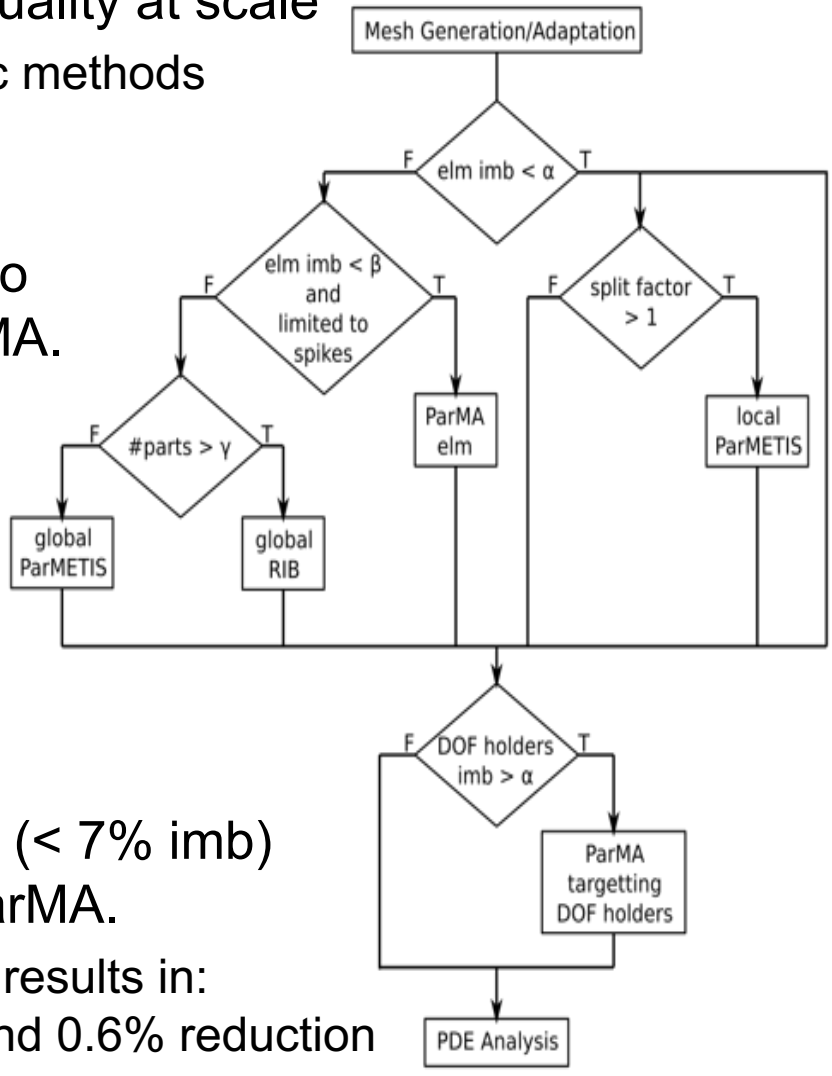
- Local and global topological and geometric methods
- ParMA quickly reduces large imbalances and improves part shape

Partitioning 1.6B element mesh from 128K to 1M parts (1.5k elms/part) then running ParMA.

- Global RIB - 103 sec, ParMA - 20 sec: 209% vtx imb reduced to 6%, elm imb up to 4%, 5.5% reduction in avg vtx per part
- Local ParMETIS - 9.0 sec, ParMA - 9.4 sec results in: 63% vtx imb reduced to 5%, 12% elm imb reduced to 4%, and 2% reduction in avg vtx per part

Partitioning 12.9B element mesh from 128K (< 7% imb) to 1Mi parts (12k elms/part) then running ParMA.

- Local ParMETIS - 60 sec, ParMA - 36 sec results in: 35% vtx imb to 5%, 11% elm imb to 5%, and 0.6% reduction in avg vtx per part



Developing Infrastructure for Select Fusion Applications

Sometimes a significant portion of the code is replaced…

Initial version of M3D-C1 lacked a sufficient mesh component

- Replaced mesh component with PUMI's
- Retained the element stiffness calculation - simply provided the needed mesh information
- Built the equation assembly routines on the PUMI mesh

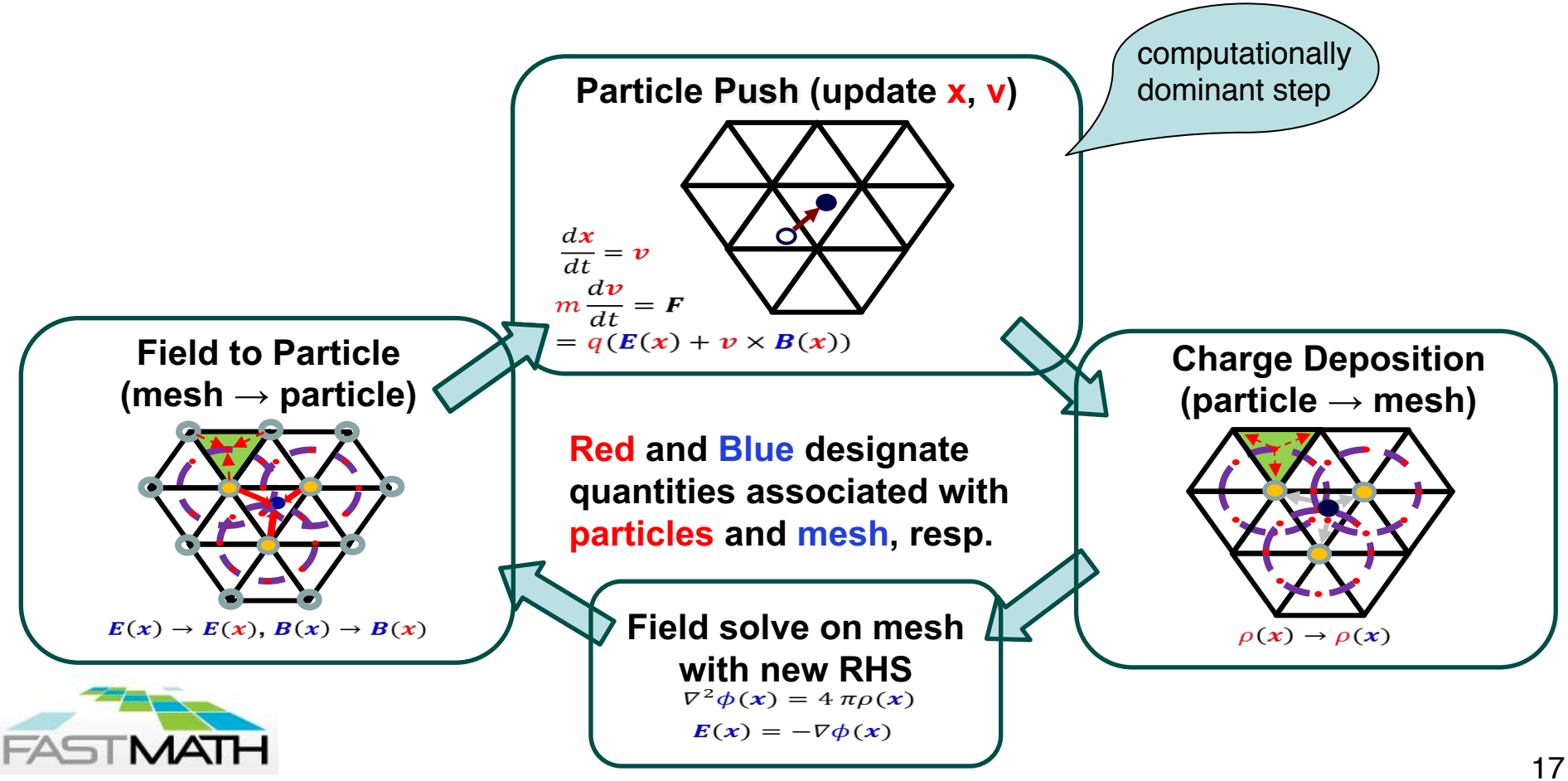
Parallel mesh version of XGC gyrokinetics PIC code

- Replaced all core mesh and particle data structures with **PUMIpic** to support particle-to-element association
- Retained the physics operations and most of the numerics
- Worked closely with XGC developers

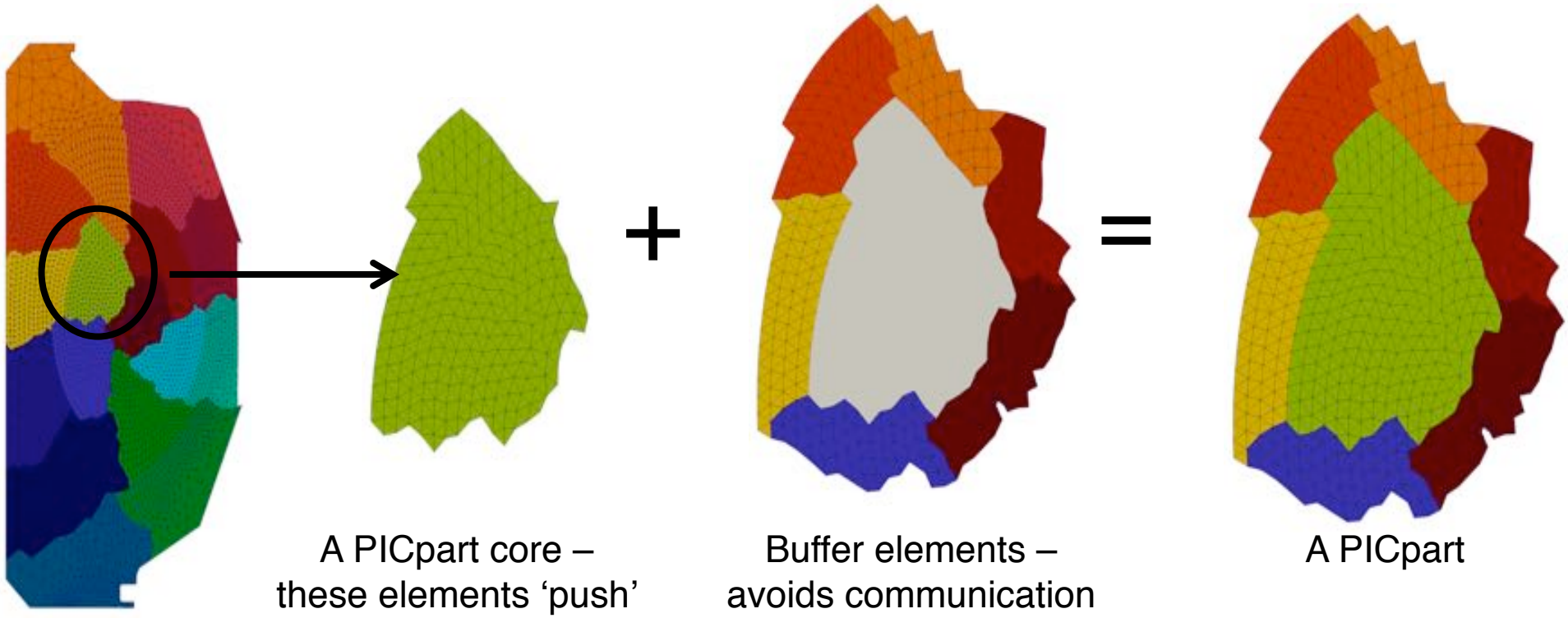
Parallel Unstructured Mesh PIC – PUMIpic

PUMIpic supports a distributed mesh

- Employ large overlaps to avoid communication during push
- All particle information accessed through the mesh



PICpart – Particle Aware Mesh Distribution



A PICpart core – these elements 'push' the particles associated with them

Buffer elements – avoids communication during computationally dominant 'push'

A PICpart

FASTMATH

Closing Remarks

- FASTMath components are designed for effective integration into application codes (in multiple ways)
- FASTMath team provides:
  - software components
  - effective means for integration into applications – there are multiple levels of capability for this
  - direct user support – mailing lists, slack, git, … direct work with application developers on their codes
- We provided our perspective of “putting it all together”

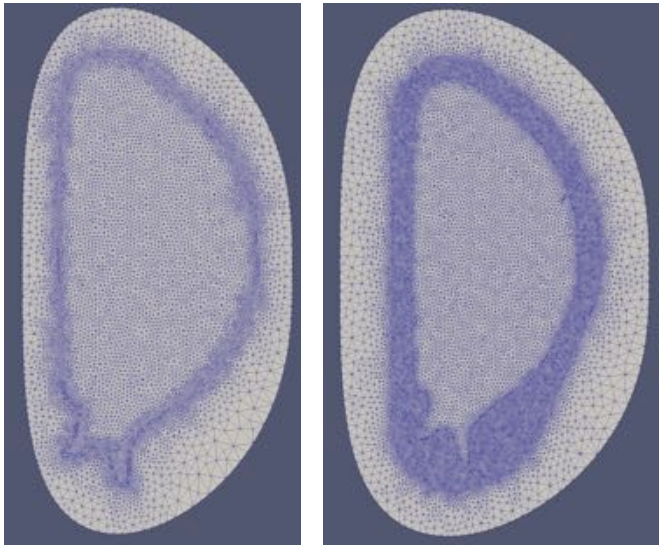
Be sure to talk with other FASTMath team members this evening on how they may help you “put it all together”

FASTMATH

Backup Slides

Developments for M3D-C1

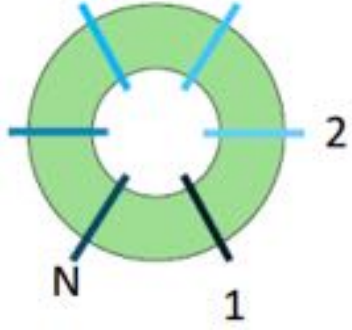
- Extended field and numbering routines to support alternative ordering of unknowns
  - By-node ordering is not ideal for numerical conditioning when the nodal dof list has derivative dof – M3D-C1 has value, 1st and 2nd derivative dof
  - Developing support for by-component ordering – all dof for the value are followed by all dof of the first derivative, etc.
- Improved solver interface (github.com/tobinw/las) toward full thread safe assembly
- Adding PIC capability to M3D-C1
  - PUMI based overlap and adjacency based element containment being used in M3D-C1 with PIC

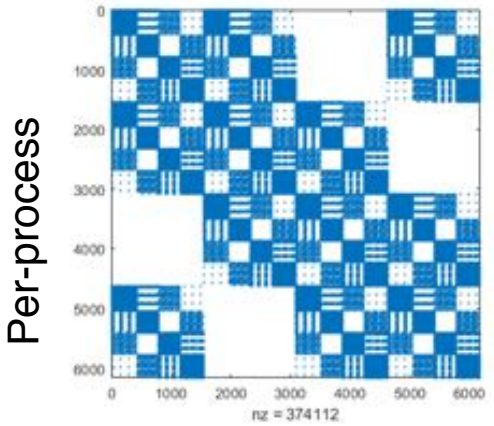


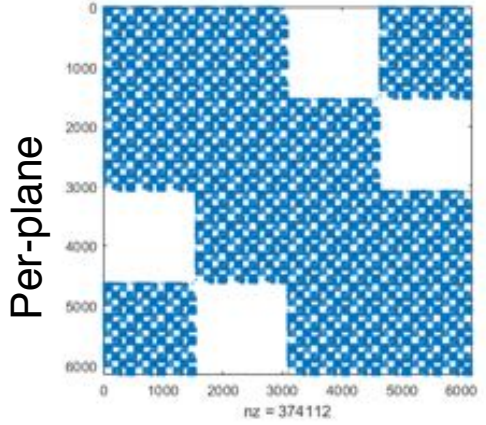
FASTMATH

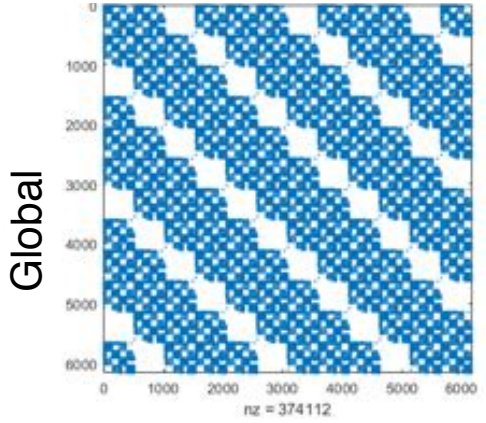
M3D-C1 By Component DOF Ordering

- Support ordering at the
  - process level,
  - poloidal plan level, or
  - globally
- Alternatives options
  - Yield different matrix sparsity patterns
  - Support different preconditioning options
  - Have different assembly interprocess communication requirements
  - Likely to yield different solution times
  - Implementation is generic – will allow the effective evaluation of the options





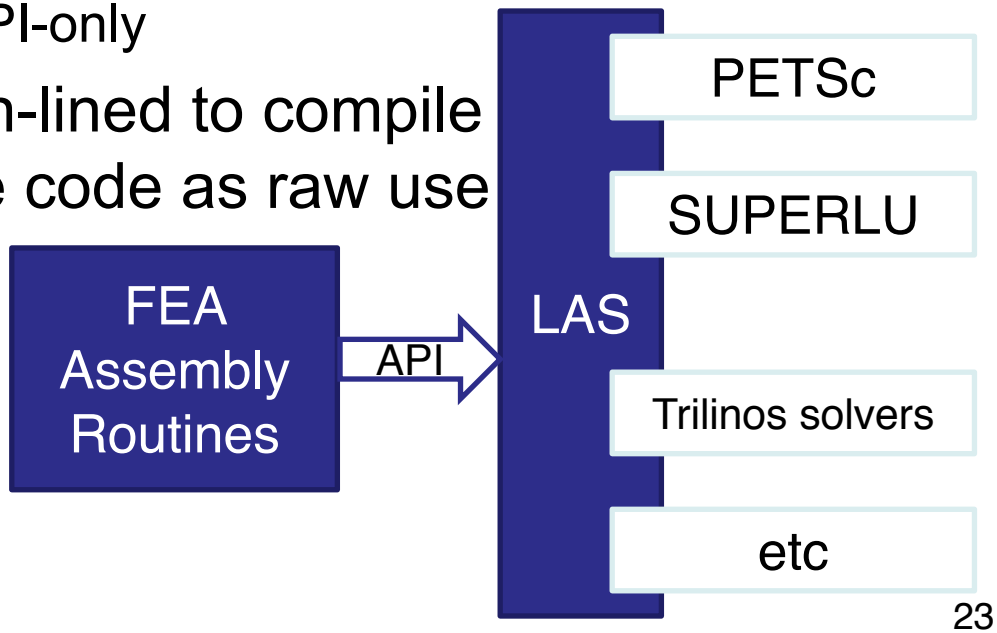




FASTMATH

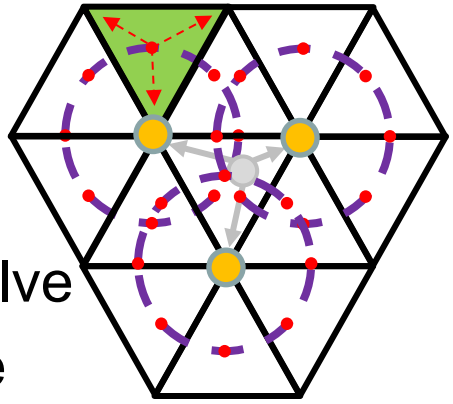
M3D-C1 Linear Solver Interface

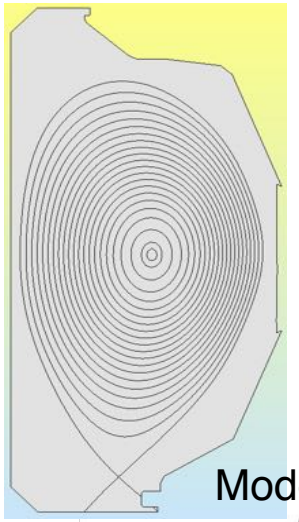
- Need more efficient and flexible linear system assembly step
- Implementing a generic linear solver interface (LAS) to wrap multiple supporting linear algebra libraries
  - Compile-time decision to target a specific backend library
    - Allows leveraging of best library/implementation for a target machine without touching matrix assembly algorithms
    - Libraries for accelerators (CUDA / PHIs)
    - Libraries for threaded or MPI-only
  - LAS API is aggressively in-lined to compile down to identical machine code as raw use of a library backend



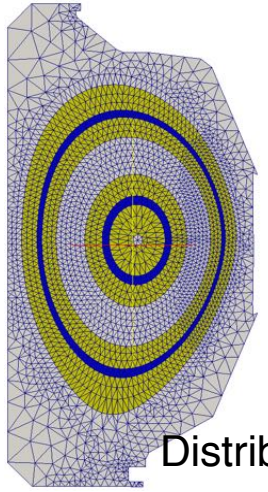
PUMIpic for XGC Gyrokinetic Code

- XGC uses a 2D poloidal plane mesh considering particle paths
  - Mesh distribution takes advantage of physics defined model/mesh
  - Separate parallel field solve on each poloidal plane
- XGC gyro-averaging for Charge-to-Mesh
- PETSc used for field solve
  - Solves on each plane
  - Mesh partitioned over N_{ranks}/N_{planes} ranks
  - Ranks for a given plane form MPI sub-communicators

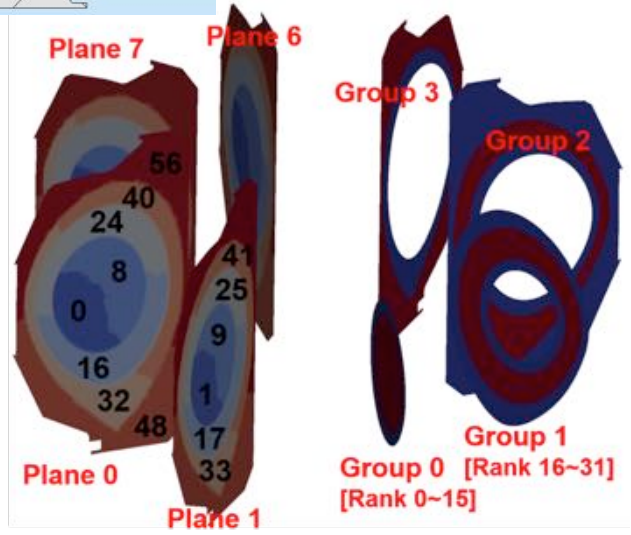




Model



Mesh Distribution



Two-level partition for solver (left) and particle push (right)

FASTMATH