FASTMath Unstructured Mesh Technologies

Presenters

Mark S. Shephard, Vijay S. Mahadevan,
Mauro Perego, Glen Hansen and Cameron W. Smith

FASTMath SciDAC Institute

Unstructured Mesh Technology and Application Team

Argonne National Laboratory

- *Vijay Mahadevan*
- Iulian Grindeanu
- Jim Jiao (Stony Brook)

Lawrence Livermore National Lab.

- Barna Bihari
- *Lori Diachin*

Sandia National Laboratories

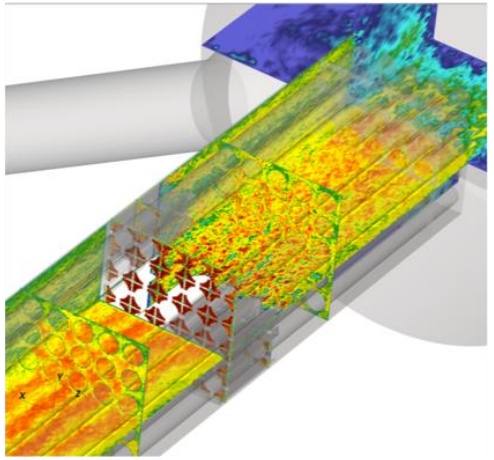
- Karen Devine
- *Glen Hansen*
- Mauro Perego
- Vitus Leung

Rensselaer Polytechnic Inst.

- Max Bloomfield
- Brian Granzow
- Dan Ibanez
- Qiukai Lu
- Onkar Sahni
- Seegyoung Seol
- *Mark Shephard*
- *Cameron Smith*
- Gerrett Diamond
- Ken Jansen (U. Colorado)
- Michel Rasquin (U. Colorado)

Presentation Outline

- Unstructured mesh methods and the need for unstructured mesh components for use by analysis code developers
- Core unstructured mesh components:
  - Parallel Mesh infrastructures
  - Mesh Generation, Adaptation, Optimization
  - Fields
  - Solution transfer
- Dynamic load balancing
- Unstructured mesh/solver developments
- Creation of parallel adaptive loops using in-memory methods
- An extendable unstructured mesh environment
- Introduction to the Hands-On Session

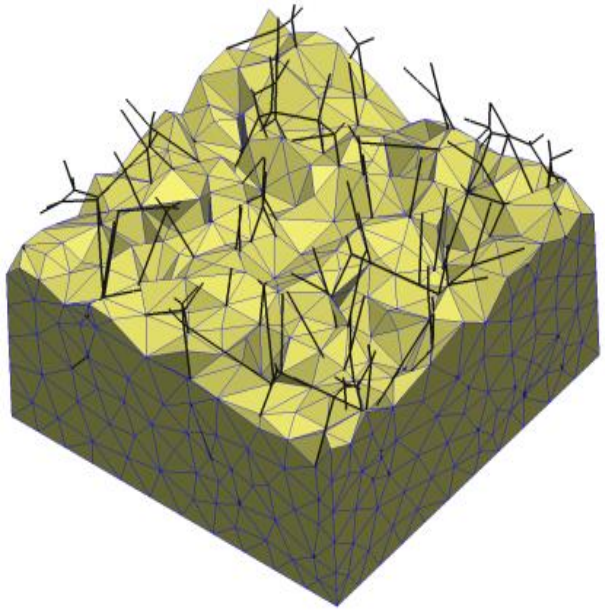


Unstructured Mesh Methods

Unstructured mesh – a spatial domain discretization composed of topological entities with general connectivity and shape

Advantages of unstructured mesh methods

- Fully automated procedures to go from CAD to valid mesh
- Can provide highly effective solutions
  - Easily fitted to geometric features
  - General mesh anisotropy to account for anisotropic physics possible
- Given a complete geometry, with analysis attributes defined on that model, the entire simulation work flow can be automated
- Meshes can easily be adaptively modified



Unstructured Mesh Methods

Disadvantages of unstructured meshes

- More complex data structures than structured meshes
  - Increased program complexity, particularly in parallel
- Can provide the highest accuracy on a per degree of freedom – requires careful method and mesh control
  - The quality of element shapes influences solution accuracy – the degree to which this happens a function of the discretization method
  - Poorly shaped elements increase condition number of global system – iterative solvers increase time to solve
  - Require careful *a priori*, and/or good *a posteriori*, mesh control to obtain good mesh configurations

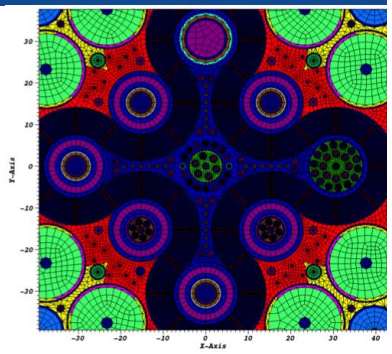
Unstructured Mesh Methods

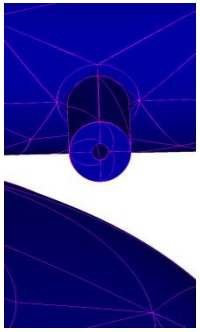
Goal of FASTMath unstructured mesh developments include:

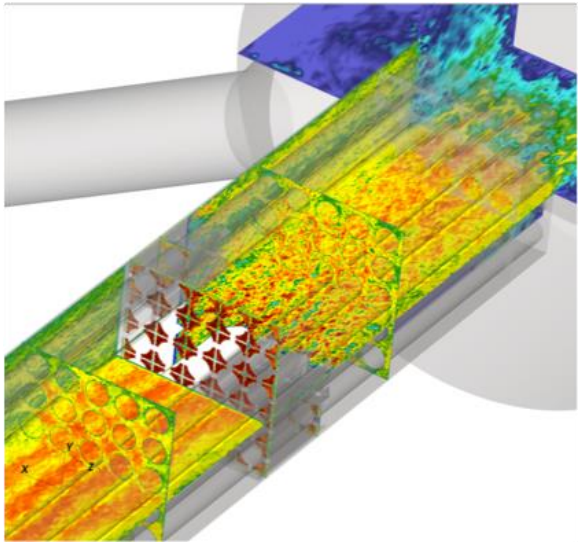
- Provide component-based tools that take full advantage of unstructured mesh methods and are easily used by analysis code developers and users
- Develop those components to operate through multi-level APIs that increase interoperability and ease of integration
- Address technical gaps by developing specific unstructured mesh tools to address needs and eliminate/minimize disadvantages of unstructured meshes
- Work with DOE applications on the integration of these technologies with their tools and to address new needs that arise

Applications using FASTMath Unstructured Mesh Components

- Accelerator Modeling (ACE3P)
- Climate data analysis (Par-NCL)
- Multi-tracer transport (MBCSLAM)
- FE-based neutron transport (PROTEUS)
- Fluid/Structure interaction (AthenaVMS)
- Fusion Edge Physics (XGC)
- Fusion Plasmas (M3DC1)
- High-order CFD on (Nektar++)
- High-speed viscous flows (FUN3D)
- Mesh-oriented FEA library (MoFEM)
- Monte Carlo neutron transport (DAG-MCNP)
- Mortar element Structural Mechanics (Diablo)
- Multiphase reactor flows (PHASTA)
- SEM-based CFD (Nek5000)
- General IM Multiphysics (Albany)
- ALE FE Shock Multiphysics (Alegra/Alexa)
- Ice sheet dynamics (Albany/FELIX)



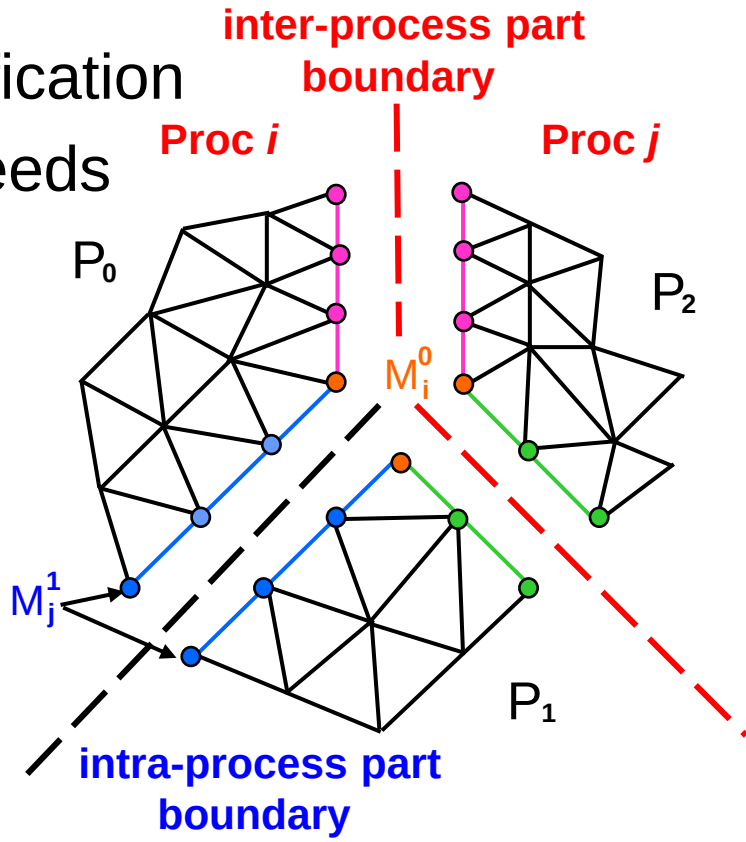




Parallel Mesh Infrastructure

Key unstructured mesh technology needed by applications

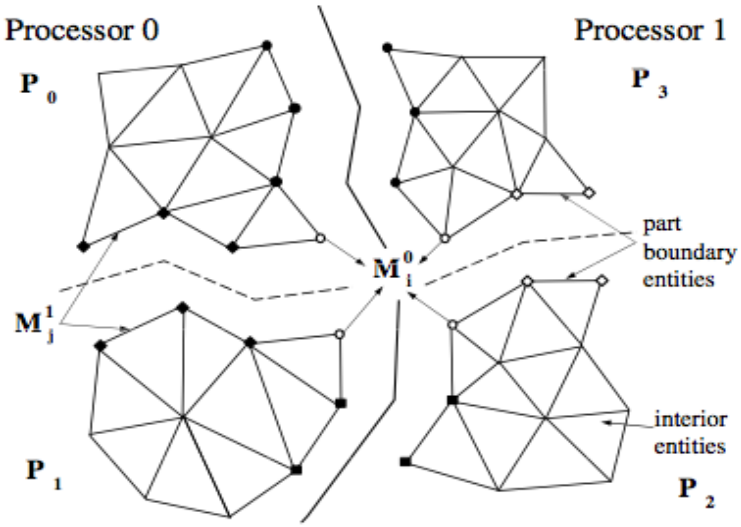
- Effective parallel mesh representation
- Base parallel functions
  - Partitioned mesh control and modification
  - Read only copies for application needs
  - Associated data, grouping, etc.
- Key services
  - Load balancing
  - Mesh-to-mesh solution transfer
  - Mesh optimization and adaptation
- Two FASTMath Implementations
  - SIGMA and PUMI



Partition Model and Migration

A ‘part’ is a set of mesh entities assigned to a process

- Treated as a serial mesh with inter-process part boundaries
- Entities on part boundaries maintain links to remote copies

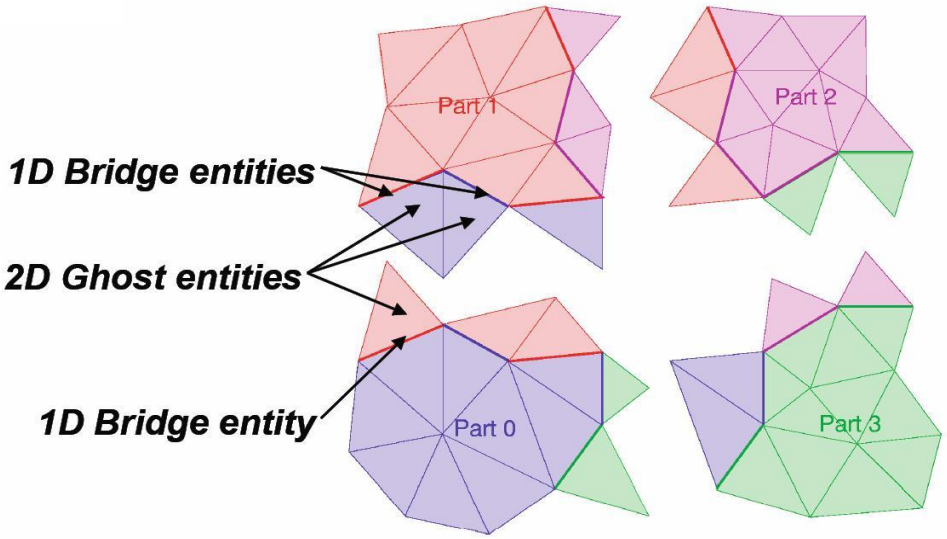


Mesh Migration

- Moving mesh entities between parts as dictated by operations
- Entities to migrate are determined based on adjacencies
- Interpart links updated based on mesh adjacencies
- Performance issues: synchronization, communications, load balance and scalability

General Functions for Read Only Copies

Copy of off-part mesh data to avoid inter-process communications



- Read-only, duplicate entity copies not on part boundary
- Copy rule: triplet (entity dim, bridge dim, # layers)
  - Entity dim: dimension for copied entities
  - Bridge dim: used to define copies through adjacency
  - # layers: # of layers measured from the part boundary
- E.g, to get two layers of region entities in the ghost layer, measured from faces on part boundary – ghost_dim=3, bridge_dim=2, and # layers=2

Mesh Generation, Adaptation and Optimization

Mesh Generation

- Must be able to create meshes over complex domains
- Already doing meshes approaching 100 billion elements
- High levels of automation needed to avoid meshing bottleneck

Mesh Adaptation must

- Use *a posteriori* information to improve mesh
- Account for curved geometry (fixed and evolving)
- Support general, and specific, anisotropic adaptation

Mesh Shape Optimization

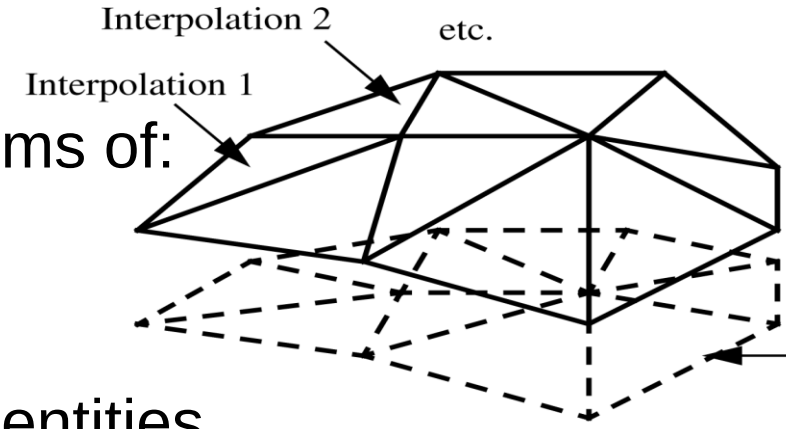
- Control element shapes as needed by the various discretization methods for maintaining accuracy and efficiency

Parallel execution of all three functions critical on large meshes

Fields and Solution Transfer

Need to support the definition of, and operations on, fields defined over space/time domains

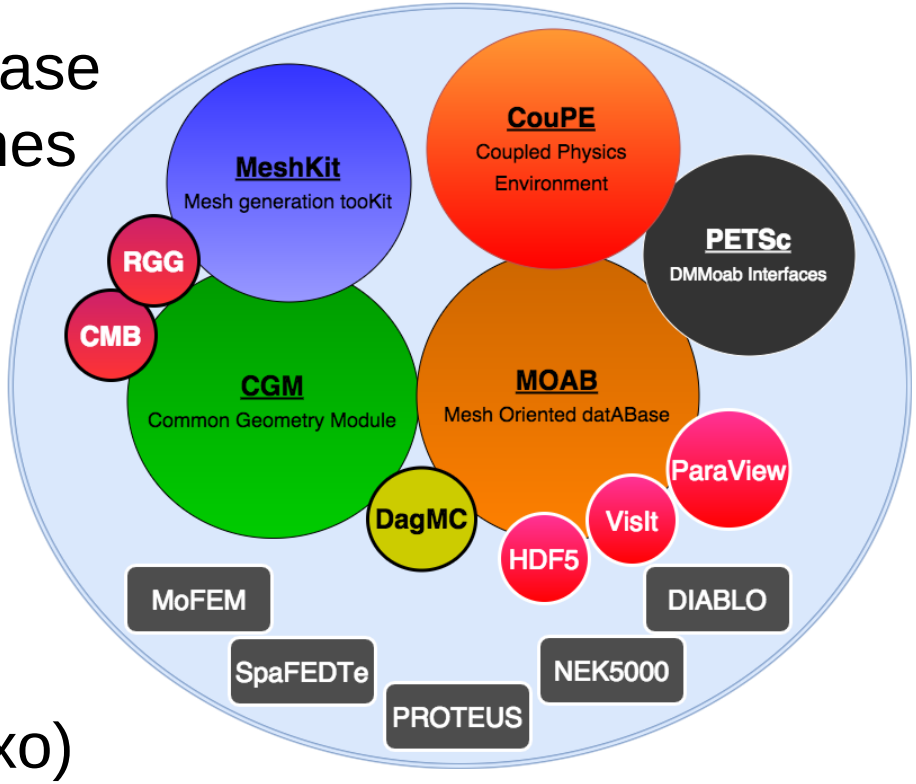
- Input fields can be defined over geometric model and meshes
- Output fields defined over meshes
- Fields are tensors and defined in terms of:
  - Tensor order and symmetries
  - Relationship to domain entities
  - Distributions of components over entities
- Must support operations on fields including:
  - Interrogations – pointwise and distributions
  - Basic – integration, differentiation, projection, etc.
  - Complex – mesh-to-mesh transfer, conservation, etc.



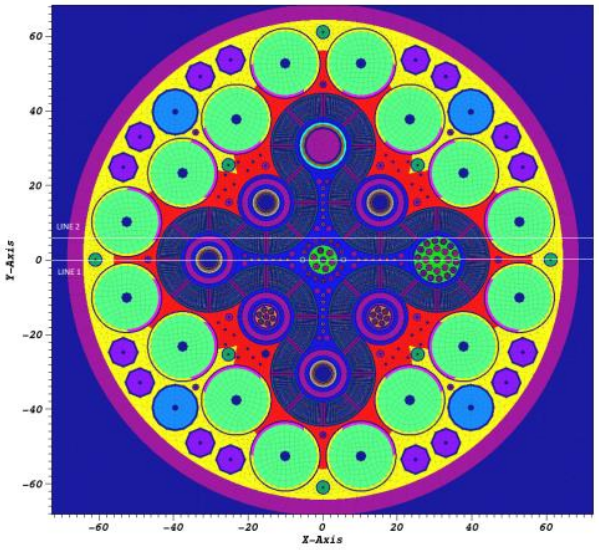
SIGMA Geometry-Mesh-Solver Interfaces

Geometry and mesh (data) generation/handling infrastructure with flexible solver interfaces. (http://sigma.mcs.anl.gov)

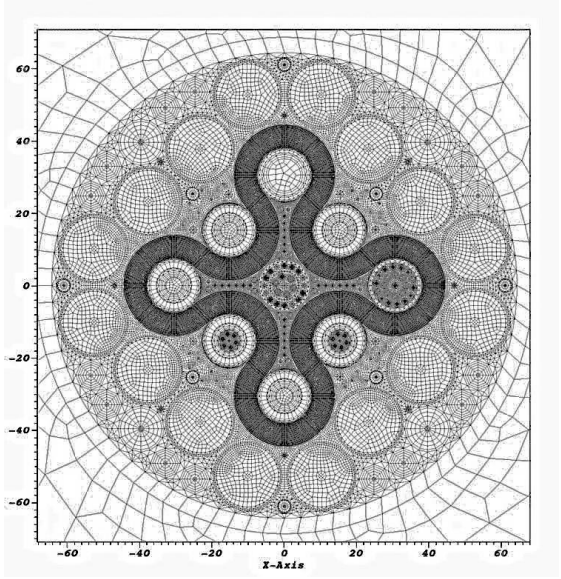
- ★ CGM – Common Geometry Module for solid engines
- ★ MeshKit – Mesh generation toolKit
- ★ MOAB – Mesh Oriented datABase for handling unstructured meshes
- ★ Solver interfaces
  - ★ PETSc – MOAB (DMMoab) Discretization Manager
  - ★ CouPE – Coupled multi-physics Environment
- ★ Scalable HDF5 serialization
- ★ In-situ visualization (h5m/vtk/exo)



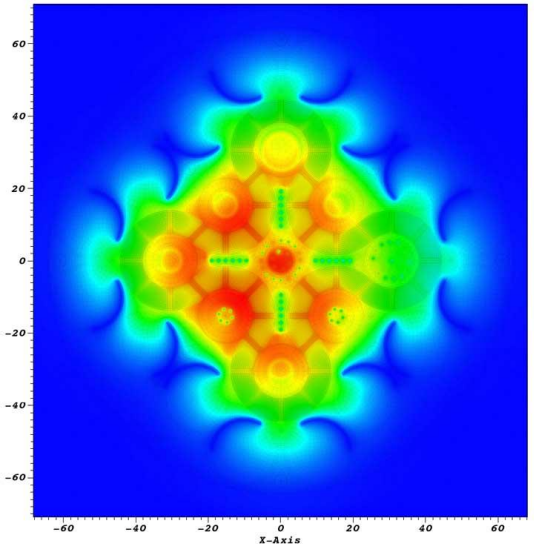
SIGMA Geometry-Mesh-Solver Interfaces



Geometry/BC Setup



Generate unstructured mesh and link with solvers



Check-point/Analyze/Visualize

- ★ CGM: Geometry abstractions to handle complex CAD models
- ★ MeshKit: Graph-based plugin design (point➔line➔quad➔hex)
- ★ Several efficient native algorithms and links to external mesh generation libraries (CUBIT, Netgen, Tetgen, CAMAL, Gmsh)
- ★ Goal: Simplified computational workflows ➔ Boost productivity

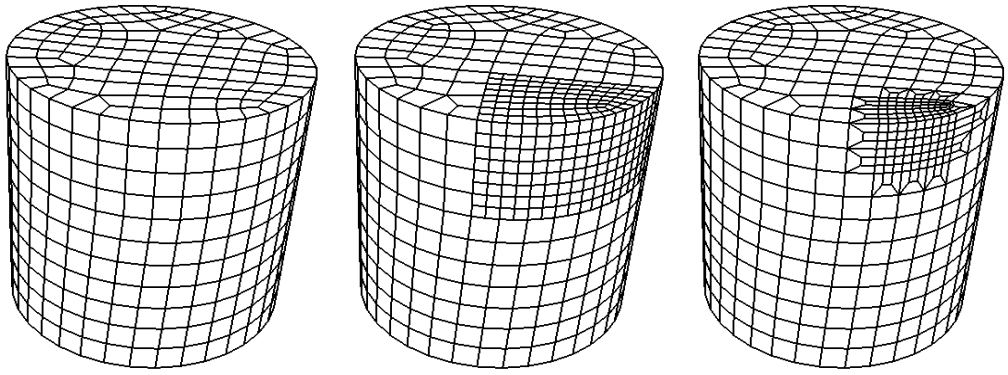
MOAB Parallel Mesh Infrastructure

- Array-based unstructured mesh data-structure
  - ★ Support stencil and block structured computations
  - ★ Support thread safety for hybrid programming models
- Dynamic load balancing: Zoltan, PARMetis
- Discretization kernels: cG, dG, Spectral, GFD, <user>
- Uniform mesh refinement hierarchy generation
  - ★ Recover high order projection: surface reconstruction
  - ★ Quantify geometry errors in absence of CAD models
- Adaptive mesh refinement
  - ★ Conformal: TRI/TET straightforward but QUAD/HEX is hard!
  - ★ Non-conformal (hanging nodes): Memory conserving designs in array-based setting is tricky.

MOAB Parallel Mesh Infrastructure

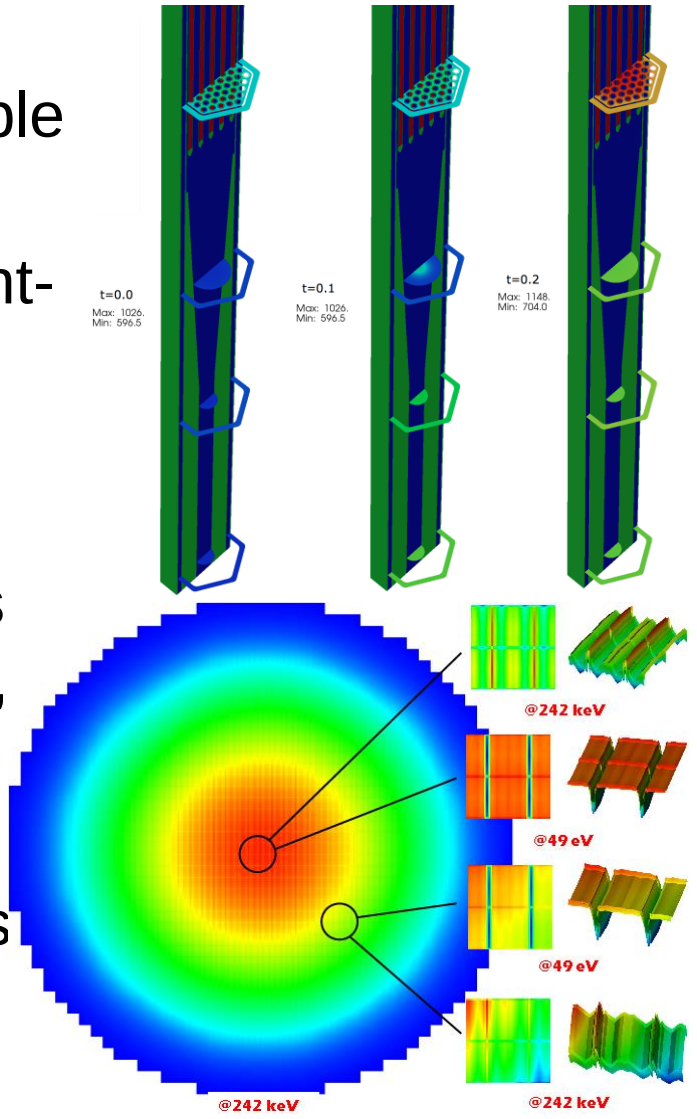
- Array-based unstructured mesh data-structure
  - ★ Support stencil and block structured computations
  - ★ Support thread safety for hybrid programming models
- Dynamic load balancing: Zoltan, PARMetis
- Discretization kernels: cG, dG, Spectral, GFD, <user>
- Uniform mesh refinement hierarchy generation
  - ★ Recover high order projection: surface reconstruction
  - ★ Quantify geometry errors in absence of CAD models
- Adaptive mesh refinement

  a) Original
  b) Non-conformal
  c) Conformal

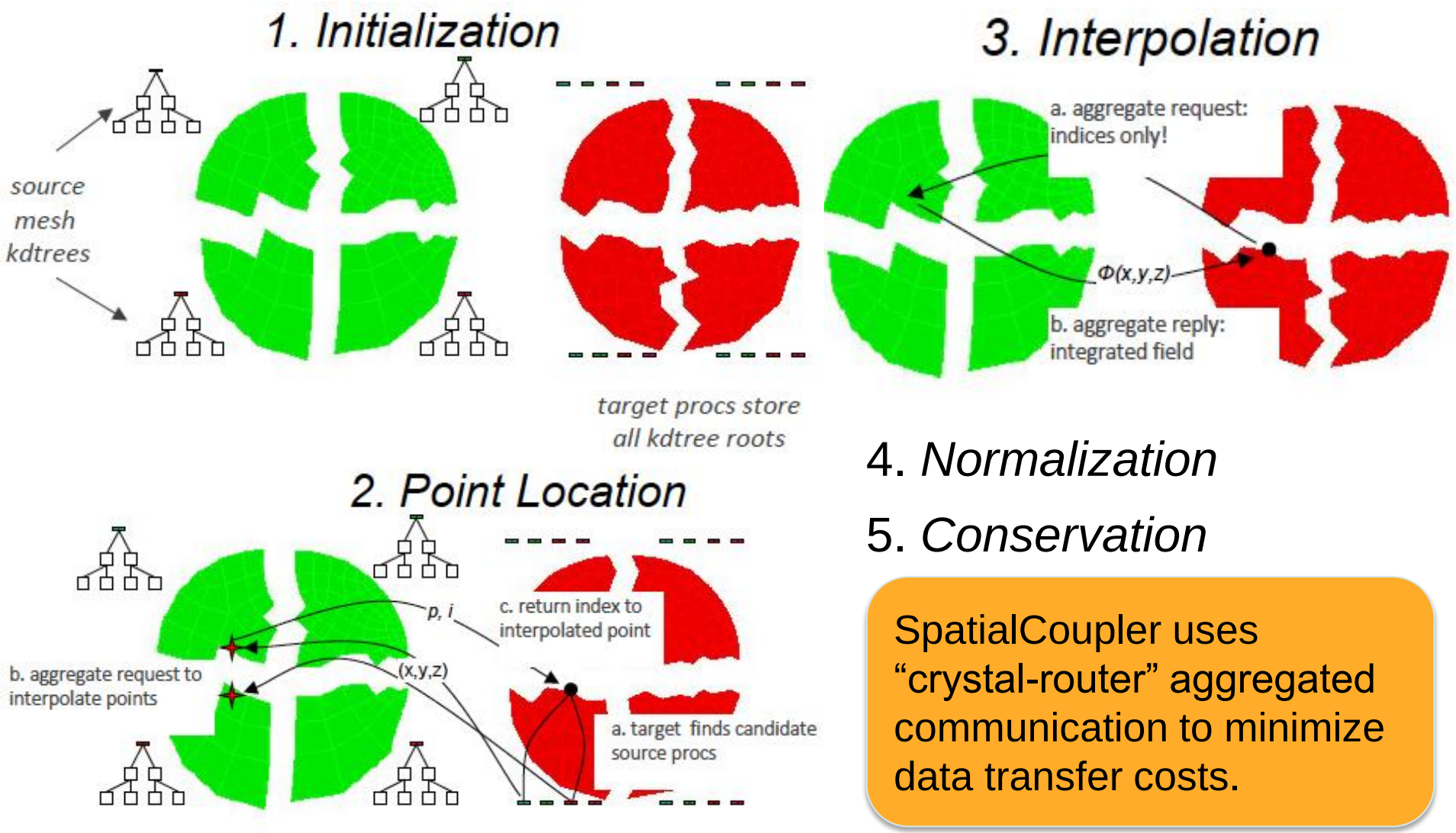


MOAB Parallel Mesh Infrastructure

- ★ Goals: Simplify geometry search and unify discretization kernels with a flexible interface for coupled problems.
- ★ Geometry search: support parallel point-in-element query for various element topologies (edge, tri/quad/polygon, tet/hex/prism/pyramid)
- ★ Discretization: support transformations higher-order basis functions (lagrange, spectral) for optimized local FE/FV
- ★ Mesh smoothing: Laplace, Lloyd, Anisotropic – for deformation problems
- ★ Other focus: Parallel I/O scalability harder than mesh manipulation!



MOAB Solution Transfer: Algorithm



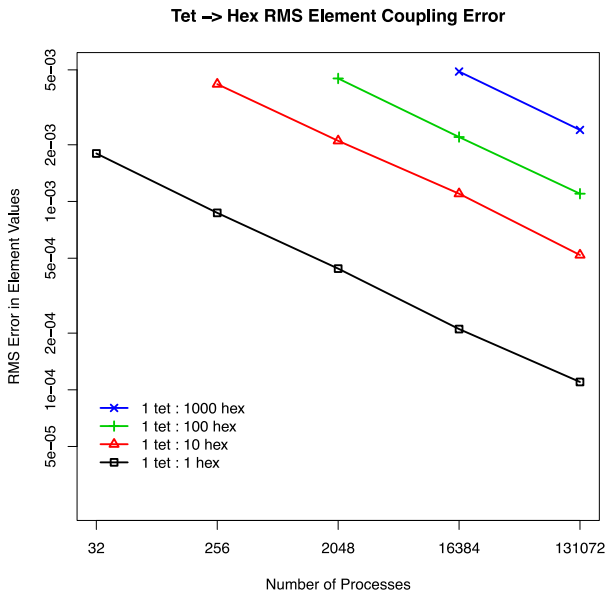
4. *Normalization*

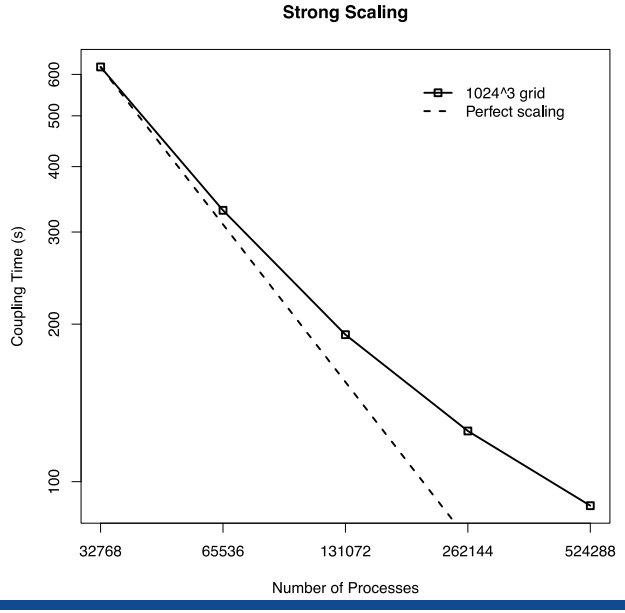
5. *Conservation*

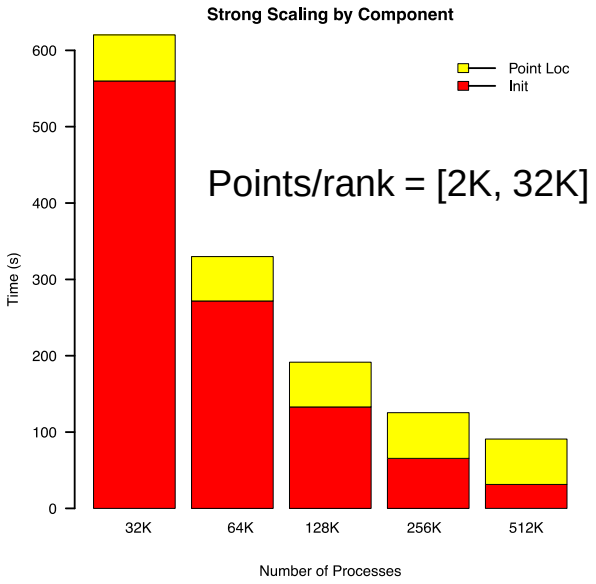
SpatialCoupler uses "crystal-router" aggregated communication to minimize data transfer costs.

MOAB Solution Transfer: Scalability

- Demonstrated 60% strong scalability of the solution transfer implementation in MOAB up to 512K cores on BG/Q.
- Bottleneck: Kd-tree scales as `O(nlog(n))`; Consider BVH/BIH trees to attain `O(log(n))` time complexity.
- Real problems: location vs interpolation, O(1) vs O(Δt)
- Initialization costs amortized over multiple interpolations!

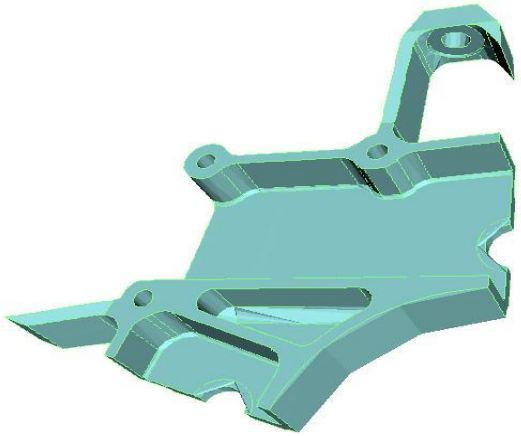




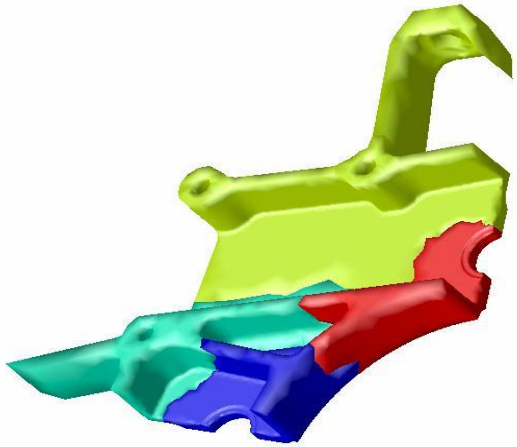


PUMI Unstructured Mesh Infrastructure

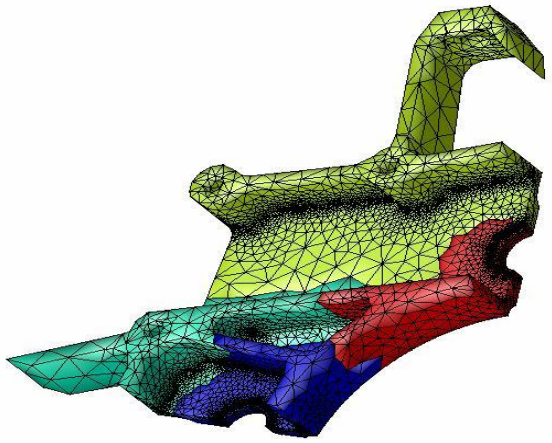
- Complete representation to provide any adjacency in O(1) time
- Array-based storage for reduced memory size
- Parallel control through partition model that supports
  - All interprocess communications
  - Effective migration of mesh entities
  - Generalized read only copies



Geometric model



Partition model

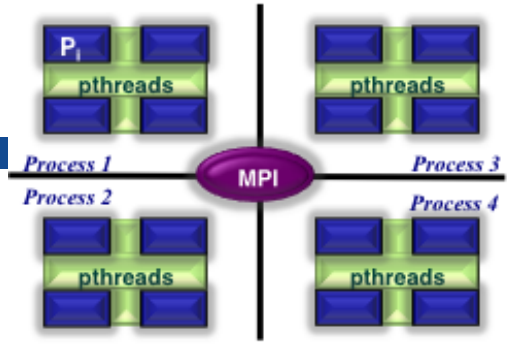


Distributed mesh

PUMI Unstructured Mesh Infrastructure

- Focused on supporting massively parallel evolving meshes as needed for adaptive mesh and/or evolving geometry problems
- Used in the construction of parallel adaptive simulation loops by combining with:
  - Fully automatic parallel mesh generation for general non-manifold domains supported by Simmetrix meshing
  - General mesh modification to adapt meshes to control discretion errors, account for evolving geometry
  - Multiple dynamic load balancing tools as needed to effectively load balance the steps in an evolving mesh simulation
- Supported evolving meshes with 92 billion elements

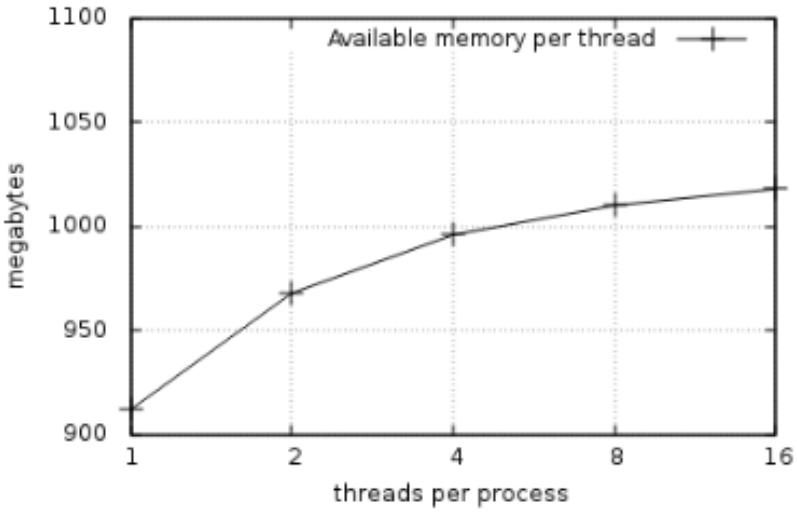
Architecture Aware PUMI



Unstructured meshes that effectively use high core-count, hybrid parallel compute nodes

- A parallel control utility (PCU) that supports hybrid threading and message passing operations on partitioned PUMI meshes
- 16 threads per process on BG/Q saves 20% of memory
  - Critical for many-core nodes where memory/core is limited

Figure 6: Available memory with threads

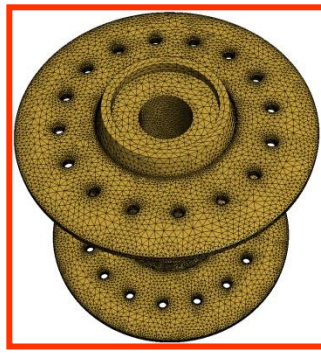


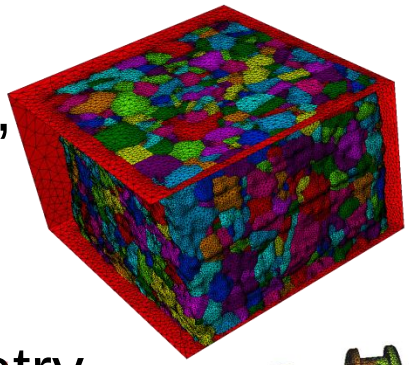
Use of Intel Phi accelerators

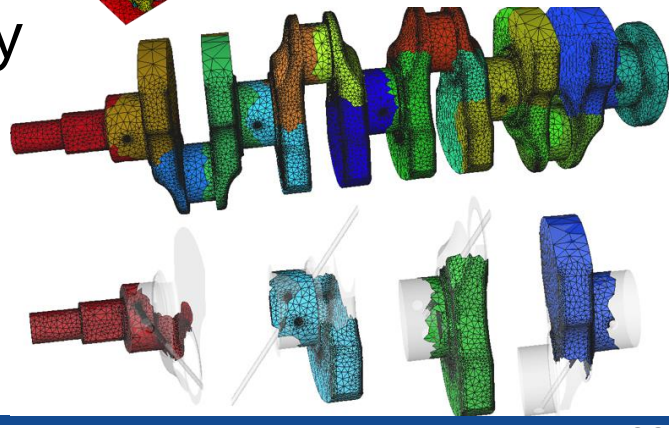
- On an equal number of Phi and BG/Q nodes
  - 1024 → 2048 partitioning is 40% faster on Stampede
  - 2048 → 4096 partitioning 8% slower on Stampede

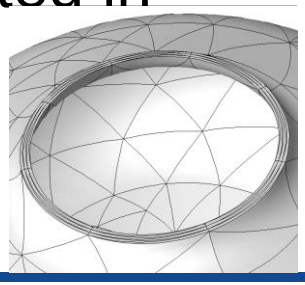
Mesh Generation

- Complete representation supports any application need
- Have made extensive use of Simmetrix meshing component
  - Any combinations of CAD and triangulations
  - Voxel (image) to model to mesh capabilities
  - Extensive control of mesh types, orders and layouts – boundary layer, anisotropic, gradation, etc.
  - Curved element meshes
  - Parallel mesh and distributed geometry
    - 1B element mesh generated in 8 minutes on 256 cores
    - 13 billion elements on up to 2048 cores







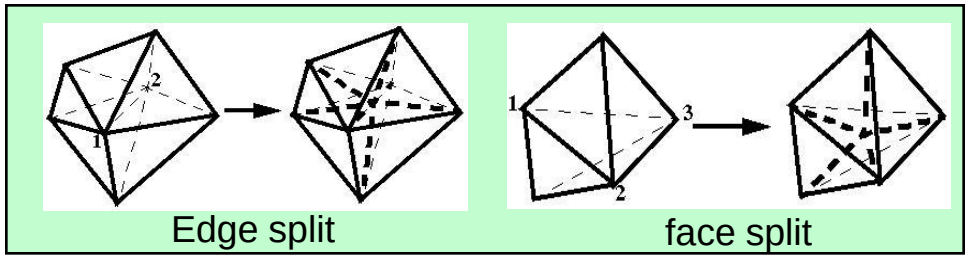


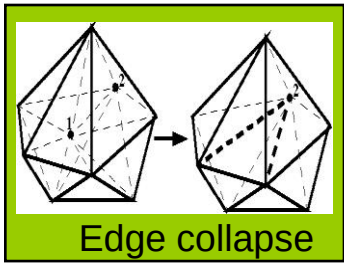
General Mesh Modification for Mesh Adaptation

- Goal is the flexibility of remeshing with added advantages
- Strategy
  - Employ a “complete set” of mesh modification operations to alter the mesh into one that matches the given mesh size field
  - Driven by an anisotropic mesh size field that can be set by any combination of criteria
- Advantages
  - Supports general anisotropic meshes
  - Can deal with any level of geometric domain complexity
  - Can obtain level of accuracy desired
  - Solution transfer can be applied incrementally - provides more control to satisfy constraints (like mass conservation)

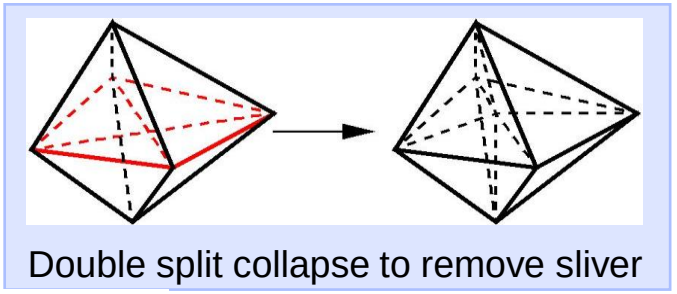
Mesh Adaptation by Local Mesh Modification

- Controlled application of mesh modification operations including dealing with curved geometries, anisotropic meshes
- Base operators
  - Swap, collapse, split, move

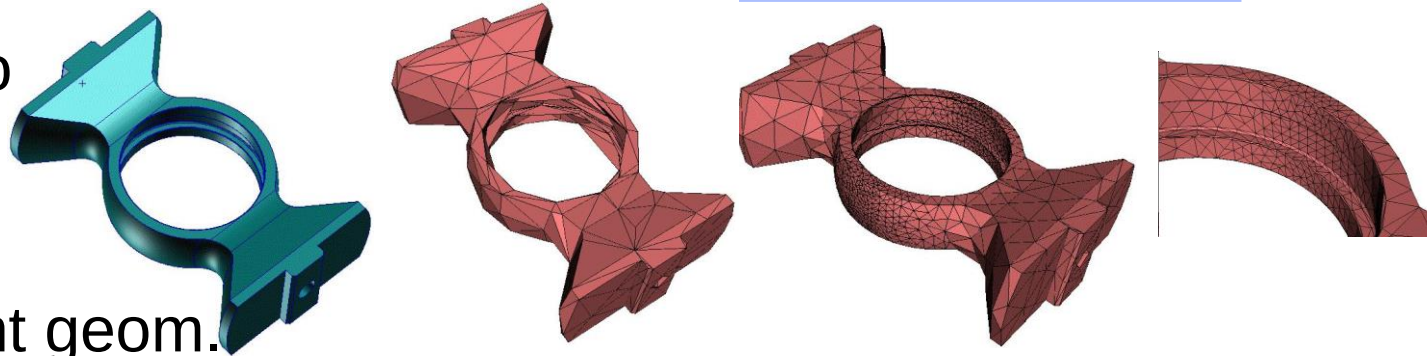




- Compound operators chain single step operators
  - Double split collapse operator
  - Swap(s) followed by collapse operator
  - Split, then move the created vertex
  - Etc.

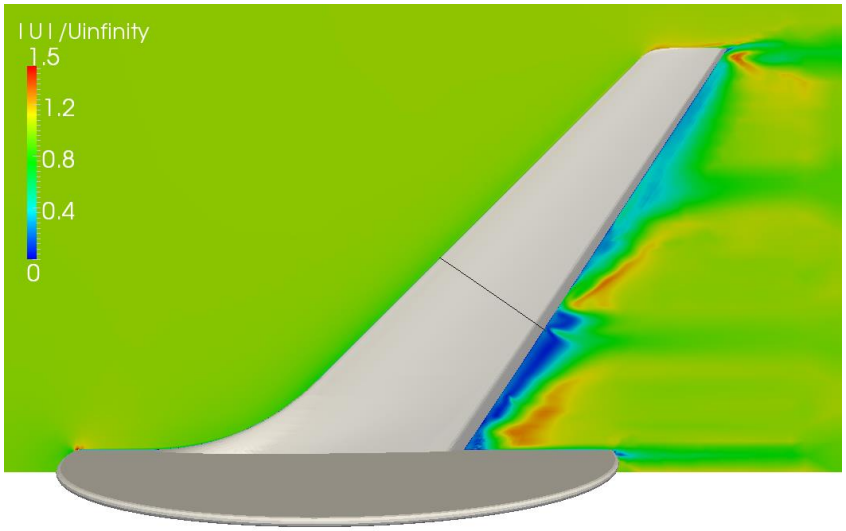


- Mesh adapts to true geometry
- Fully parallel
- Curved element geom.



Attached Parallel Fields (APF)

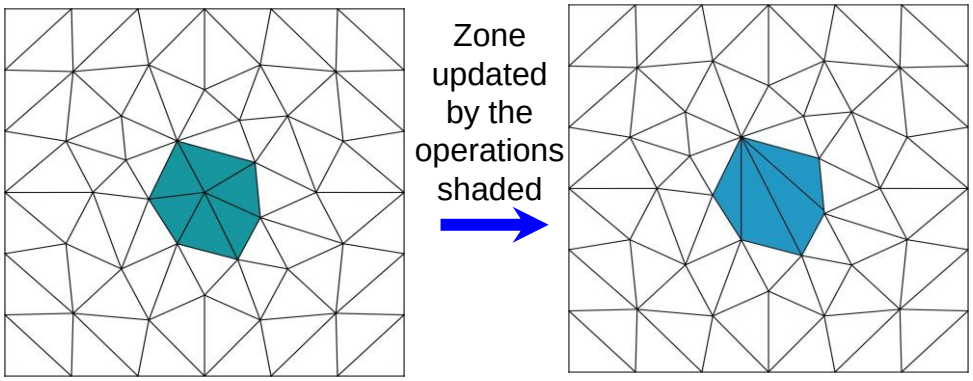
- Attached Parallel Fields (APF)
- Effective storage of solution fields on meshes
- Supports operations on the fields
  - Interrogation
  - Differentiation
  - Integration
  - Interpolation/projection
- Recent efforts
  - Adaptive expansion of Fields from 2D to 3D in M3D-C1
  - History-dependent integration point fields for Albany plasticity models



Local Solution Transfer

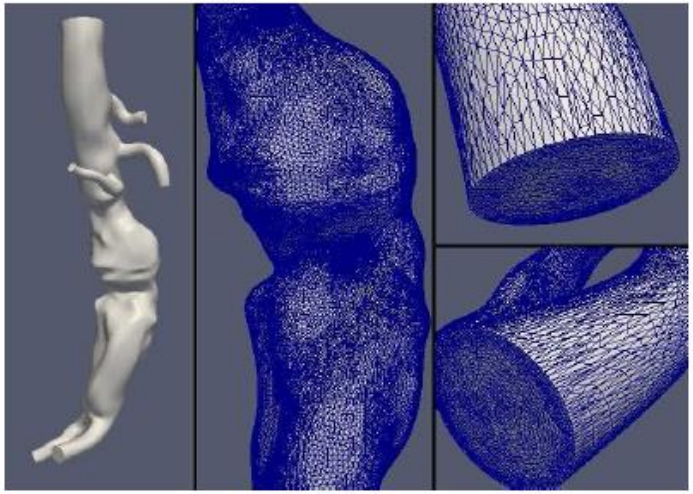
Local solution transfer during mesh adaptation

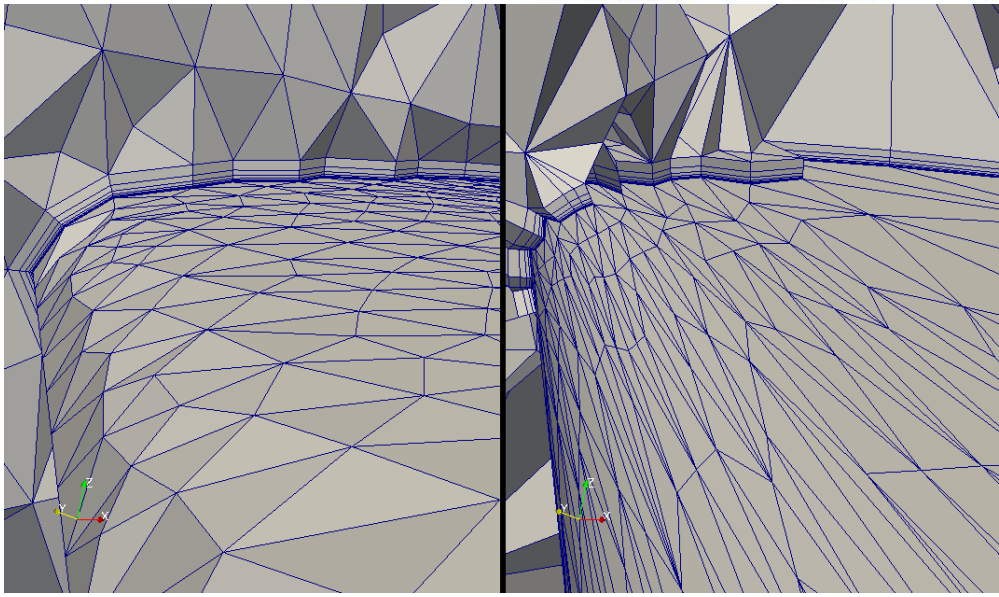
- Performed on cavity as local mesh modification performed
- Limited number of elements involved (no search over mesh)
- No accuracy loss with some operations (e.g., refinement)
- Others easier to control due to local nature (e.g., more accurate conservation correction)
- Applied to primary & secondary variables in multiple applications
- In the metal forming case not only was the transfer faster, the non-linear solve was much faster since "equilibrium recovery" iterations not required

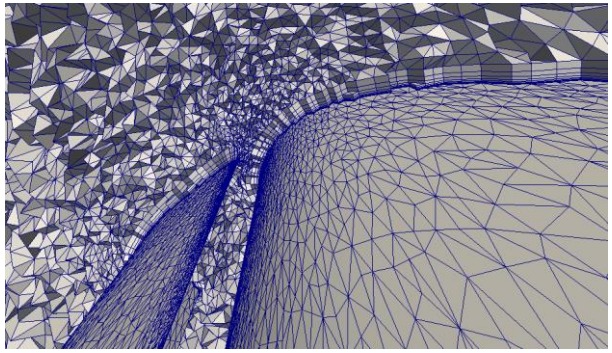


Mesh Adaptation Status

- Applied to very large scale models – 3.1M processes on ¾ million cores
- Local solution transfer supported through callback
- Effective storage of solution fields on meshes
- Supports adaptation with boundary layer meshes

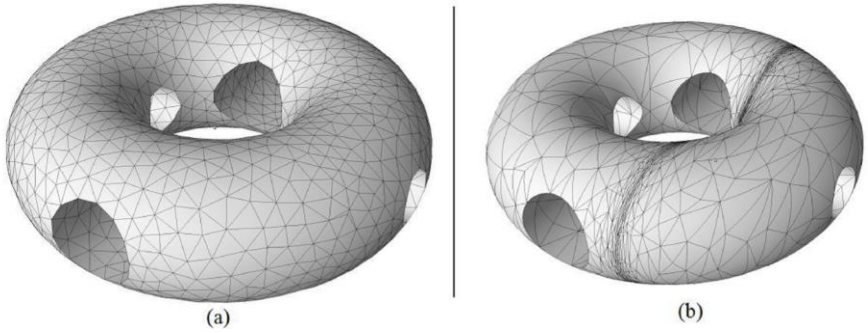


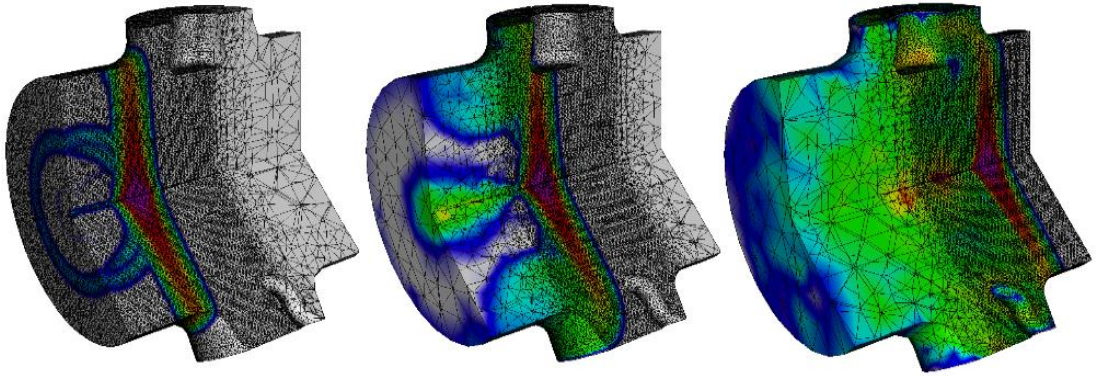


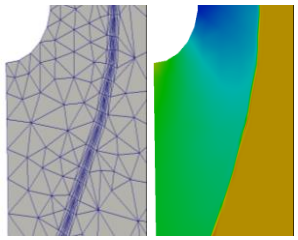


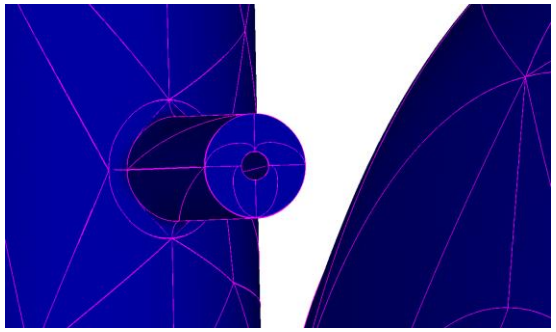
Mesh Adaptation Status

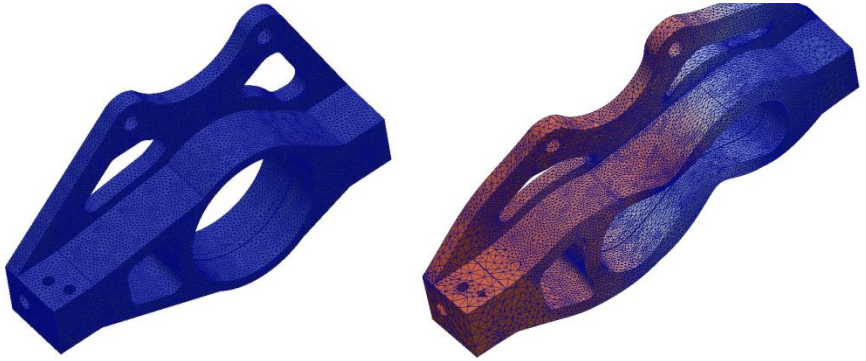
- Supports adaptation of curved elements
- Adaptation based on multiple criteria, examples
  - Level sets at interfaces
  - Tracking particles
  - Discretization errors
  - Controlling element shape in evolving geometry











Highlight: Unstructured Mesh Infrastructure for the M3D-C1 MHD Code for Fusion Plasma Simulations

- Provide the mesh infrastructure for M3D-C1
  - Geometric model interface defined by analytic expressions with B-splines
  - Distributed mesh management including
    - process grouping to define plane
    - each plane loaded with the same distributed 2D mesh then
    - 3D mesh and corresponding partitioning topology constructed
  - Mesh adaptation and load balancing
  - Adjacency-based node ordering
  - Mapping of mesh to PETSc structures and control of assembly processes

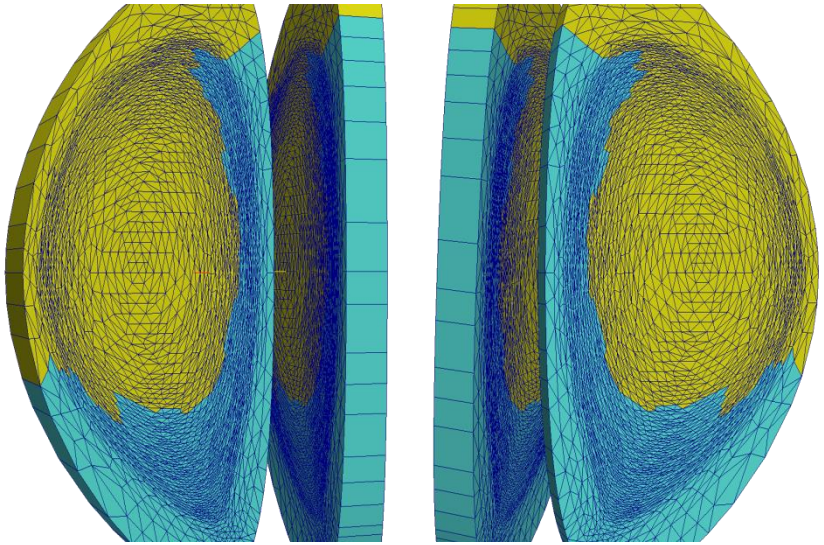
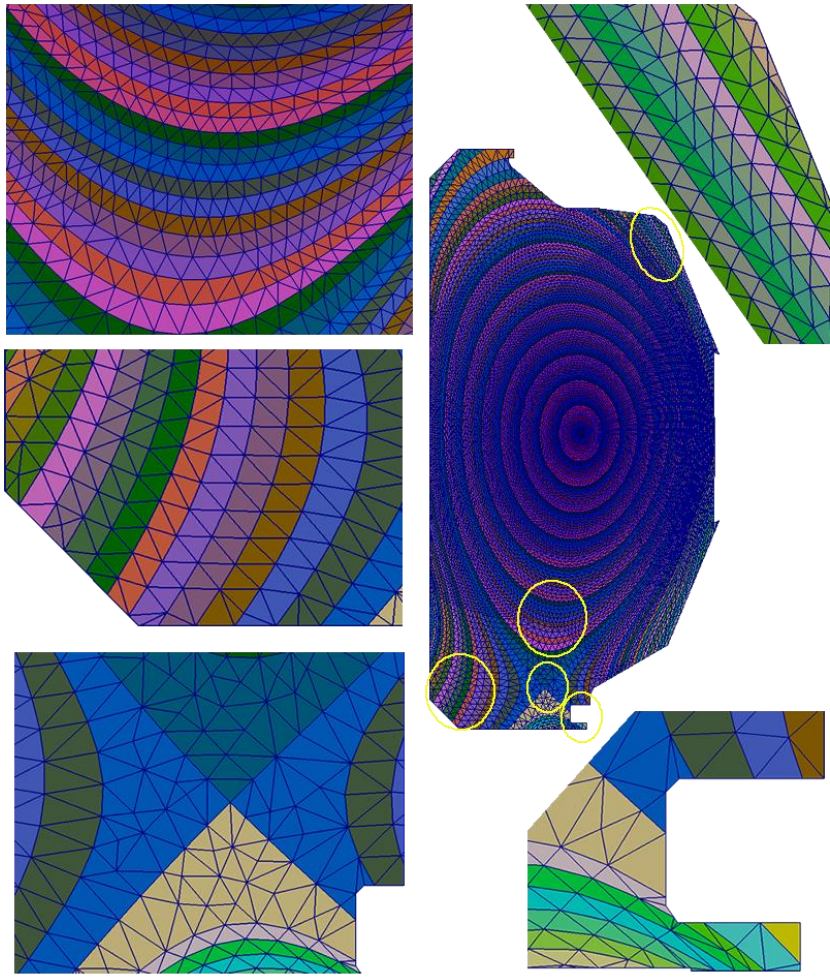


Fig: 3D mesh constructed from 64 2D planes on 12288 processes [1] (only the mesh between selected planes shown)

[1] S.C.Jardin, et al, Multiple timescale calculations of sawteeth and other macroscopic dynamics of tokamak plasmas, Computational Science and Discovery 5 (2012) 014002

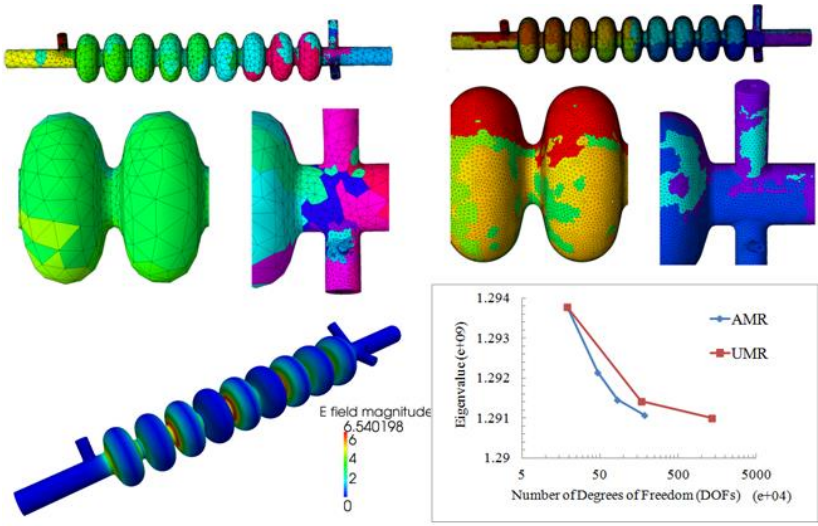
Highlight: Unstructured Mesh Techniques for Edge Plasma Fusion Simulations

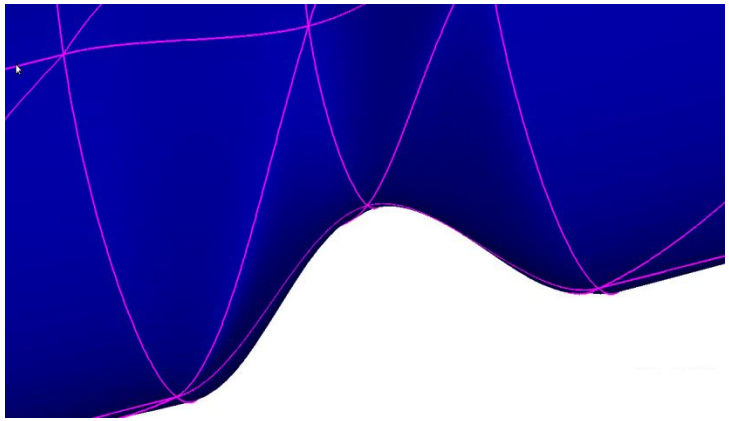
- EPSI PIC coupled to mesh simulation requires high quality meshes meeting a strict set of layout and other constraints
  - Previous method took >11 hours and mesh did not have desired quality
  - FASTMath meshing technologies put together to produce better quality meshes that meet constraints
  - Controlled meshes now generated in minutes
- Particle-in-Cell with distributed mesh
  - Current XGC copies entire mesh on each process
  - PUMI distributed mesh being extended to support parallel mesh with particles than can move through the mesh



Highlight: Parallel Mesh Adaptation with Curved Mesh Geometry for High-Order Accelerator EM Simulations

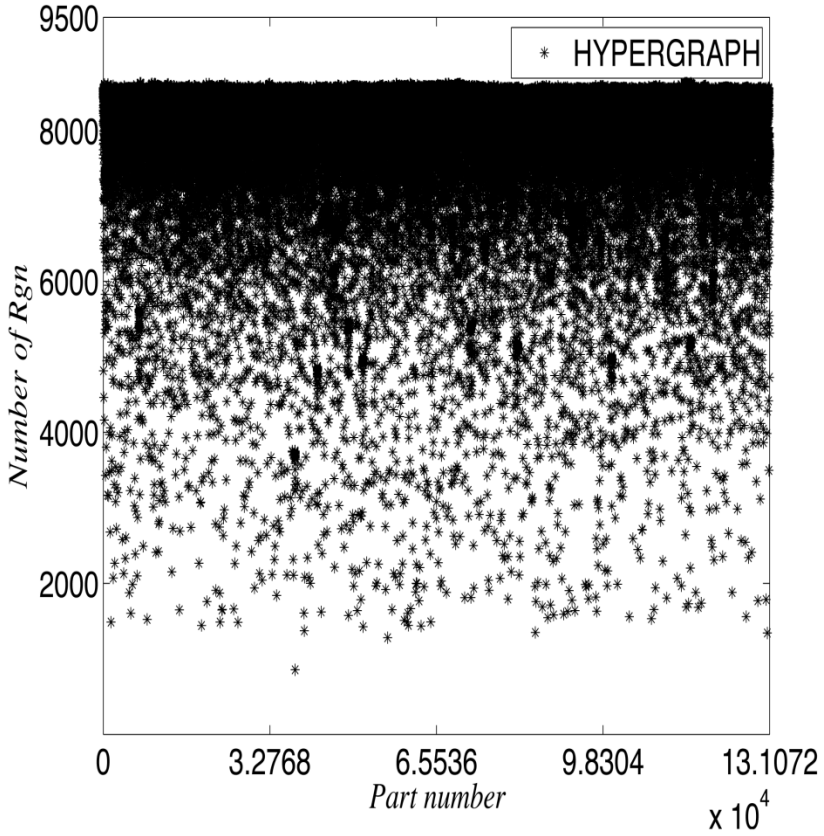
- Provide parallel mesh modification procedure capable of creating/adapting curved mesh geometry
- Parallel mesh adaptation procedure developed that supports quadratic curved meshes
- Ongoing efforts to support higher order G1 mesh geometry
- The procedure integrated with high-order electro-magnetic solver, ACE3P from the SLAC National Accelerator Laboratory



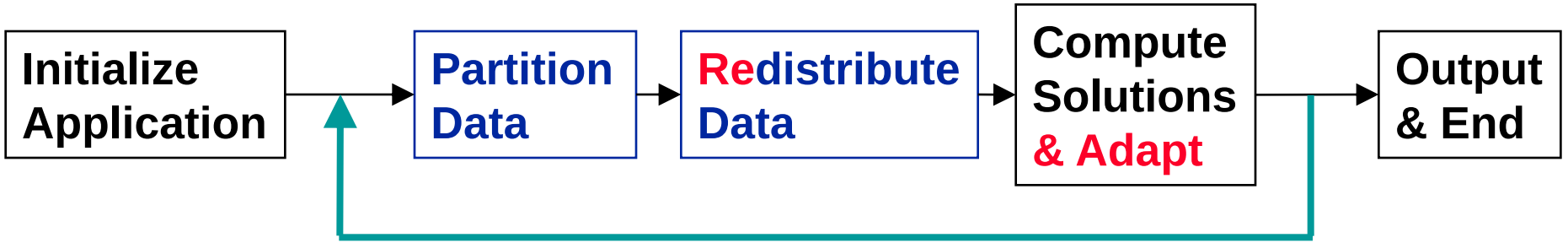


Dynamic Load Balancing

- Purpose: to rebalance load during mesh modification and before each key step in the parallel workflow
  - Equal "work load" with minimum inter-process communications
- FASTMATH load balancing tools
  - Zoltan/Zoltan2 libraries provide multiple dynamic partitioners with general control of partition objects and weights
  - ParMA – Partitioning using mesh adjacencies
  - ParMA and Zoltan2 can use each other's methods



Dynamic Load Balancing



- Dynamic repartitioning (load balancing) in an application:
  - Data partition is computed.
  - Data are distributed according to partition map.
  - Application computes and, perhaps, adapts.
  - Process repeats until the application is done.
- Ideal partition:
  - Processor idle time is minimized.
  - Inter-processor communication costs are kept low.
  - Cost to redistribute data is also kept low.

Sandia National Laboratories

Static vs. Dynamic: Usage and Implementation

Static:

- Pre-processor to application.
- Can be implemented serially.
- May be slow, expensive.
- File-based interface acceptable.
- No consideration of existing decomposition required.

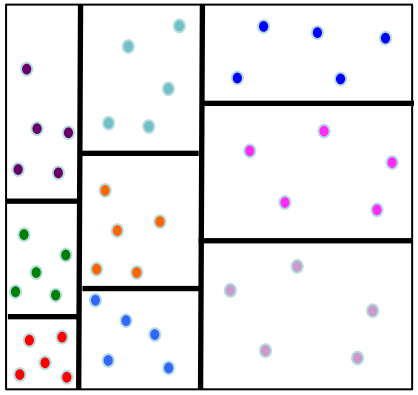
Dynamic:

- Must run side-by-side with application.
- Must be implemented in parallel.
- Must be fast and scale.
- Library application interface required.
- Should be easy to use.
- Incremental algorithms preferred.
  - Small changes in input result in small changes in partitions.
  - Explicit or implicit incrementally acceptable.

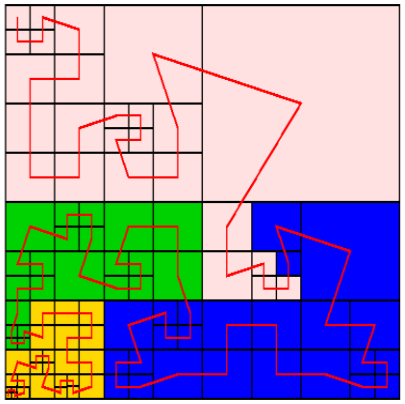
Zoltan/Zoltan2 Toolkits: Partitioners

Suite of partitioners supports a wide range of applications; no single partitioner is best for all applications.

Geometric

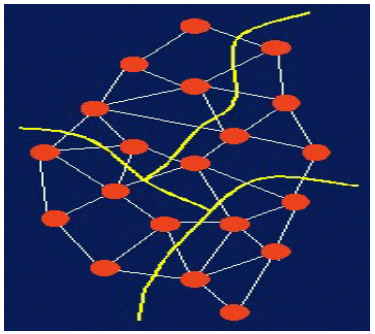


Recursive Coordinate Bisection
Recursive Inertial Bisection
Multi-Jagged Multi-section

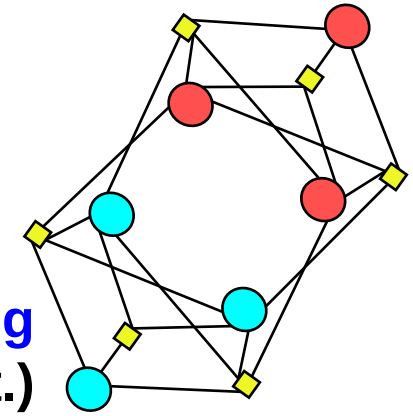


Space Filling Curves

Topology-based



PHG Graph Partitioning
Interface to ParMETIS (U. Minnesota)
Interface to PT-Scotch (U. Bordeaux)



PHG Hypergraph Partitioning
Interface to PaToH (Ohio St.)

Geometric Partitioners in Zoltan/Zoltan2

Goal: Create parts containing physically close data

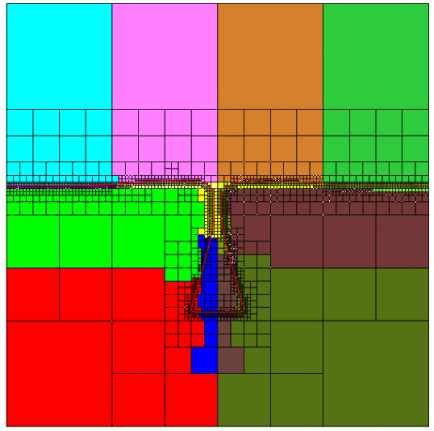
- RCB/RIB: Compute cutting planes that recursively bisect workloads
- MJ: Multi-section instead of bisection to reduce cost of partitioning
- SFC: Partition linear ordering given by space-filling curve

Advantages:

- Conceptually simple; fast and inexpensive
- Effective when connectivity info is not available (e.g., in particle methods)
- Enable efficient searches for contact detection, particle methods
- RCB/MJ: Regular parts useful in structured or unstructured meshes on elongated domains
- SFC: Linear ordering may improve cache performance

Disadvantages:

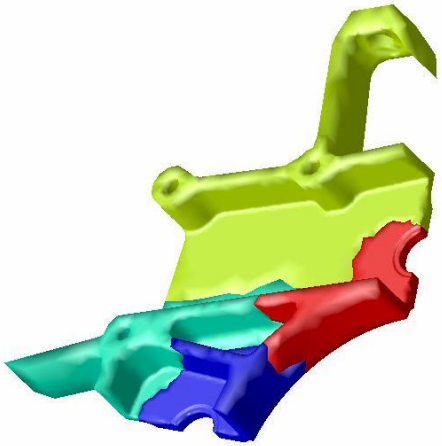
- No explicit control of communication costs
- Geometric coordinates needed



Topology-based Partitioners

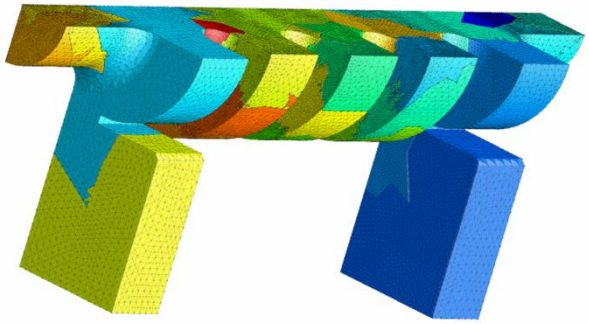
Goal: Balance work while minimizing data dependencies between parts

- Represent data with vertices of graph/hypergraph
- Represent dependencies with graph/hypergraph edges



Advantages:

- High quality partitions for many applications
- Explicit control of communication costs
- Available tools
  - Serial: Chaco, METIS, Scotch, PaToH, Mondriaan
  - Parallel: Zoltan, ParMETIS, PT-Scotch, Jostle



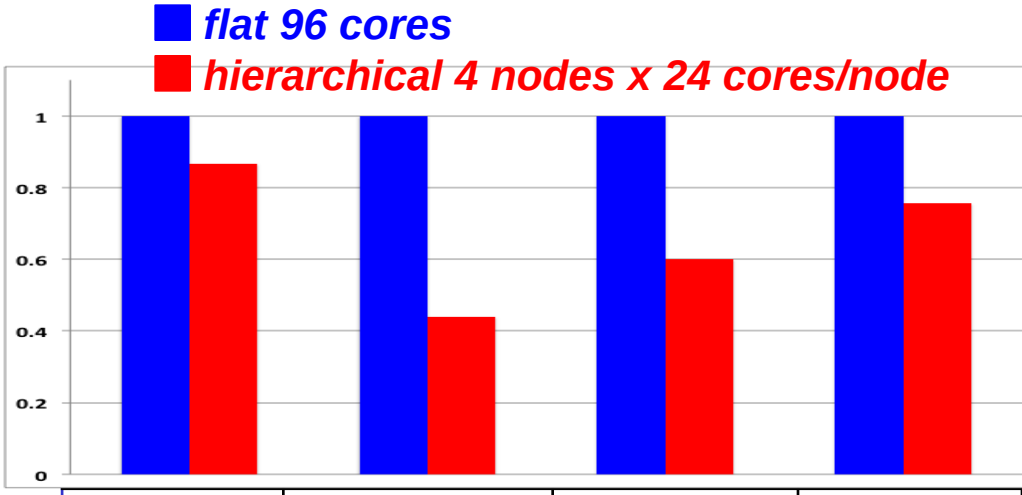
Disadvantages:

- More expensive than geometric approaches
- Require explicit dependence info

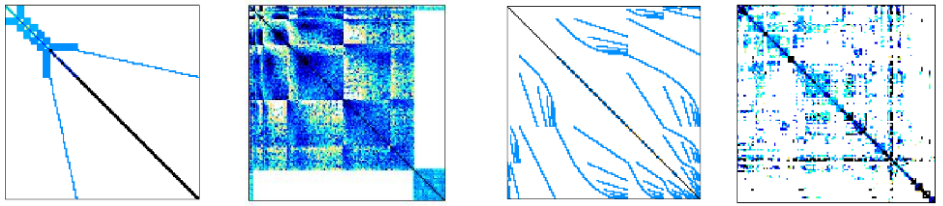
Hierarchical Partitioning in Zoltan

- Partition with respect to the machine hierarchy
  - Network, nodes, cores
  - Improved data locality in each level
- Example: Matrix-vector multiplication with 96 parts on Hopper
  - Reduced matvec time by partitioning with respect to nodes, then cores

Matvec time normalized wrt flat 96-part partition



	G3-Circuit	Thermo-mech_TC	Parabolic_FEM	Bmw7st_1
#rows	1.6M	102K	526K	141K
#nonzeros	7.7M	712K	3.7M	7.3M



Partitioning using Mesh Adjacencies (ParMA)

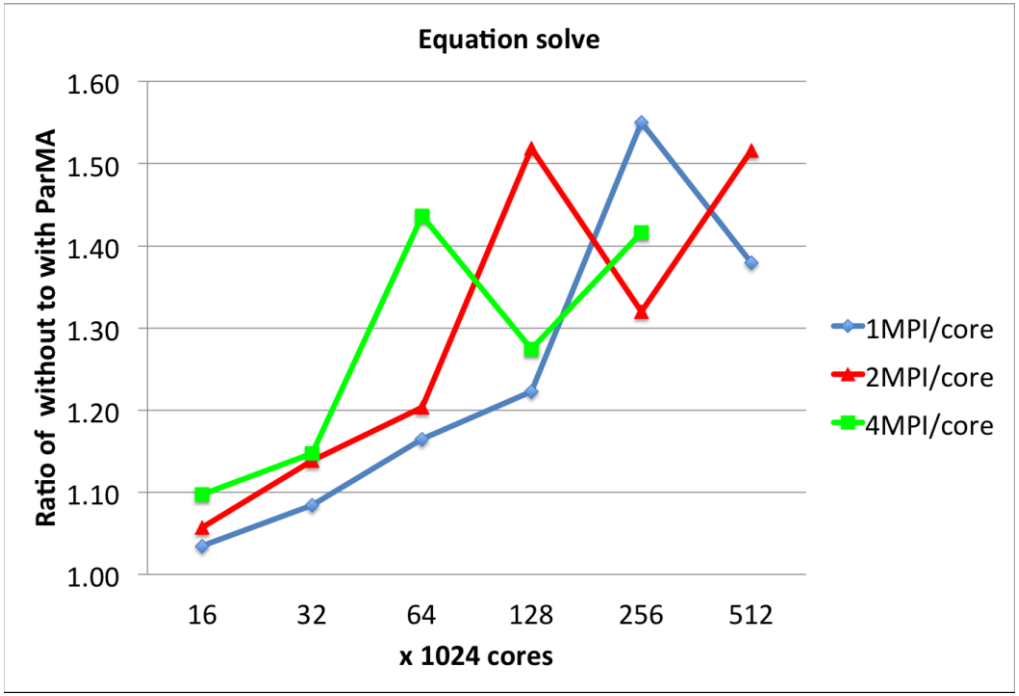
Mesh and partition model adjacencies directly used

- Directly account for multiple entity types – important for the solve process – most computationally expensive step
- Avoid graph construction
- Easy to use with diffusive procedures
- Algorithm: From high to low priority if separated by '>' and From low to high dimension entity types if separated by '='
  - (1) Compute the migration schedule. (2) Select regions for migration. (3) Migrate the selected regions.
- Partition improvement applications to date
  - Account for multiple entity types and cost functions – improved scalability of solvers
  - Support meshes with billions of elements on up to 1M cores

ParMA Application Partition Improvement

Example of C0, linear shape function finite elements

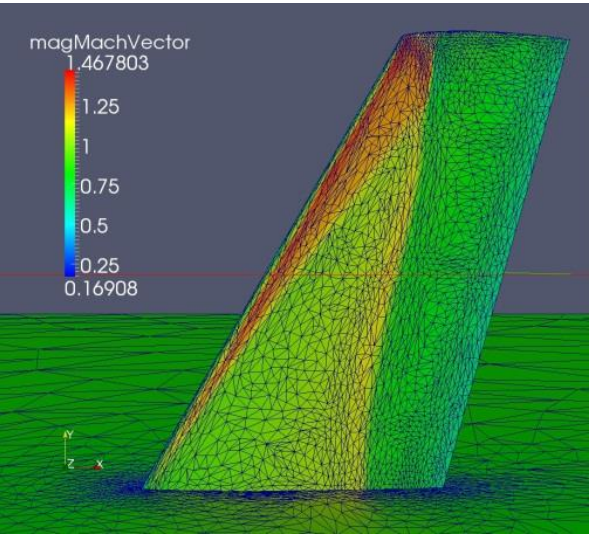
- Assembly sensitive to mesh element imbalances
- Solve sensitive to vertex imbalances - they hold the dof
  - Heaviest loaded part dictates solver performance
- Element-based partitioning results in spikes of dofs
- ParMA diffusion reduces equation solution time in PHASTA CFD by 52% on 1M cores
  - Elm imb. 11% to 4%
  - Vtx imb. 86% to 6%

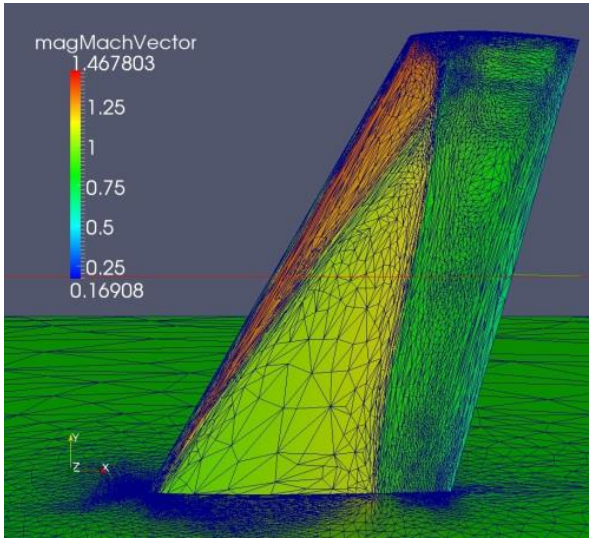


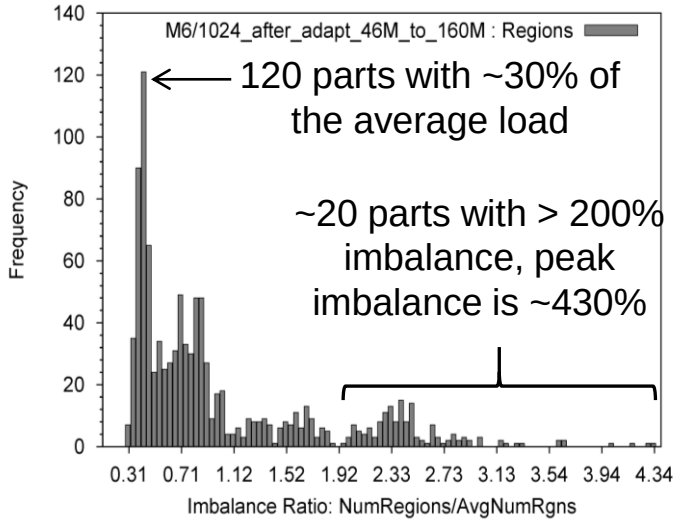
Improvement of PHASTA performance with ParMA.

Predictive Load Balancing

- Mesh modification before load balancing can lead to memory problems - common to see 400% increase on some parts
- Employ predictive load balancing to avoid the problem
  - Assign weights based on what will be refined/coarsened
  - Apply dynamic load balancing using those weights
  - Perform mesh modifications



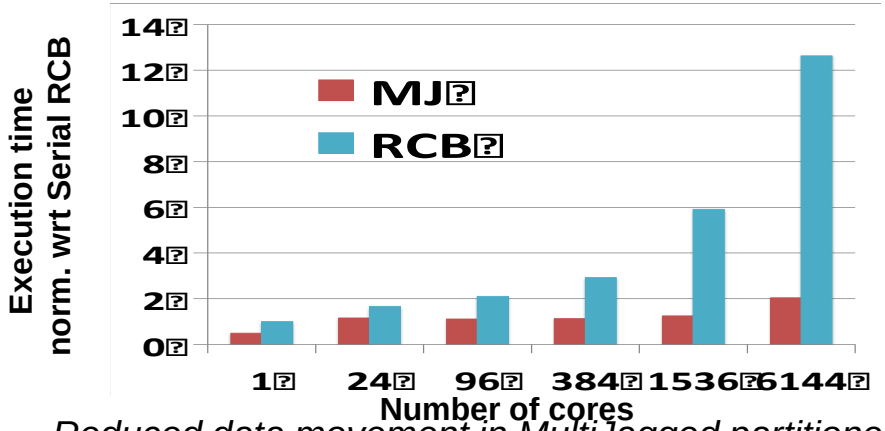




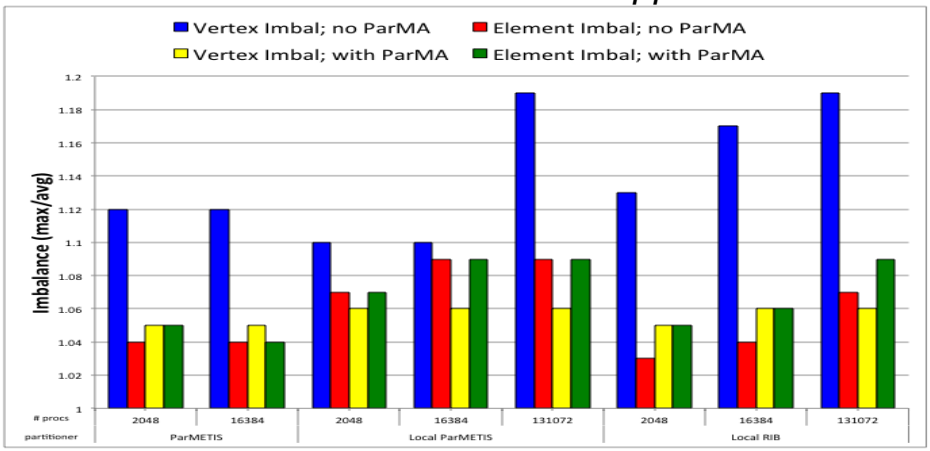
Histogram of element imbalance in 1024 part adapted mesh on Onera M6 wing if no balancing applied prior to adaptation.

Dynamic Load Balancing to Ensure the Ability of Applications to Achieve and Maintain Scalability

- Results/Impact
  - Zoltan2's MJ provides scalable partitioning on up to 524K cores in multigrid solver MueLu
  - ParMA improves PHASTA CFD code scaling by balancing multiple entity types
  - Predictive load balancing increases performance of parallel mesh adaptation
  - Multi-level/multi-method partitioning enables partitioning of 92B-element mesh to 3.1M parts on ¾ million cores



Reduced data movement in MultiJagged partitioner enables better scaling than Recursive Coordinate Bisection on NERSC's Hopper.



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

Architecture-Aware Task Placement in Zoltan2

Goal: Assign MPI tasks to cores so that application communication costs are low

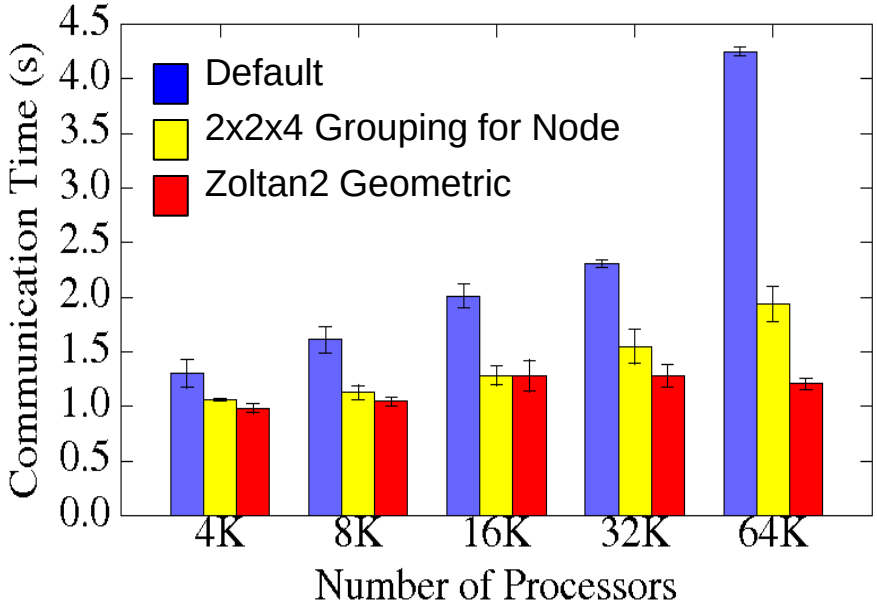
- Especially important in non-contiguous node allocations (e.g., Hopper Cray XE6)

Approach: Use Zoltan2's MJ geometric partitioner to map interdependent tasks to "nearby" cores in the allocation

- Using geometric proximity as a proxy for communication cost

Example: Task Placement in Finite Difference Mini-app MiniGhost (Barrett et al.)

- Communication pattern: 7-pt stencil
- Mapping methods:
  - None: default linear task layout (first in x, then y, then z)
  - 2x2x4 Grouping: accounts for Cielo's 16 core/node architecture
  - Geometric: also accounts for proximity of allocated nodes in network
- On 64K cores of Cielo, geometric mapping reduced MiniGhost execution time
  - by 34% on average relative to default
  - by 24% relative to node only grouping



Unstructured Mesh/Solver Developments

Need to effectively integrate parallel mesh infrastructures with unstructured mesh analysis codes

- Two key steps in unstructured mesh analysis codes
  - Evaluation of element level contributions – easily supported with FASTMath partitioned mesh infrastructures support mesh level information including link to geometry
  - Formation and solution of the global equations – interactions needed here are more complex with multiple alternatives

Two FASTMath activities related to mesh/solver interactions

- MOAB-based Discretization Manager (DM) linked with the PETSc solver library
- PHASTA massively parallel unstructured mesh code including integration with PETSC

MOAB Discretization Manager

Uniform interfaces to solve multi-component problems with FD/FEM/FVM, on both structured and unstructured meshes.

- A native MOAB implementation that exposes the underlying array-based mesh data structures through the DM (Discretization Manager) object in PETSc (DMMoab)
- Discretize the physics PDE described on MOAB mesh while leveraging the scalability of PETSc solvers.
- Build meshes in-memory for simple geometries (Cube/ Cylinder/Sphere) or load an unstructured grid from file.
- Solve and analyze efficient unstructured mesh traversal, FD/FEM-type operator assembly for relevant multi-dimensional, multi-component problems.

MOAB Discretization Manager

- Design follows structured (DMDA) and unstructured (DMPlex) interfaces; software productivity.
- Support both strided and interleaved access of field components; Opens up better preconditioning strategies.
- Provide a uniform interface to solve nonlinear problems with FEM/FDM on both structured and unstructured meshes.
- Dimension-independent operator assembly routines
- Capabilities to define field components, manage degrees-of-freedom, local-to-global transformations.
- Optimized physics residual computation using PETSc Vec that reuses contiguous memory provided by MOAB tags.
- Reduce total memory usage by sharing vector spaces and allowing block filling of coupled component terms.

MOAB Discretization Manager – Examples

http://www.mcs.anl.gov/petsc/petsc-current/docs/manualpages/DM/index.html

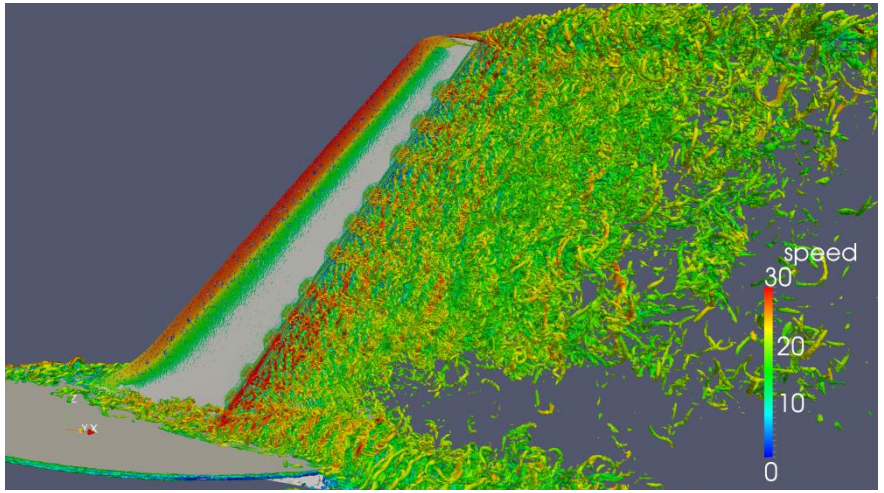
- Some relevant tutorial examples in PETSc:
  - ★ 2-D/3-D, verifiable Diffusion-Reaction FEM steady state solver with geometric multigrid.

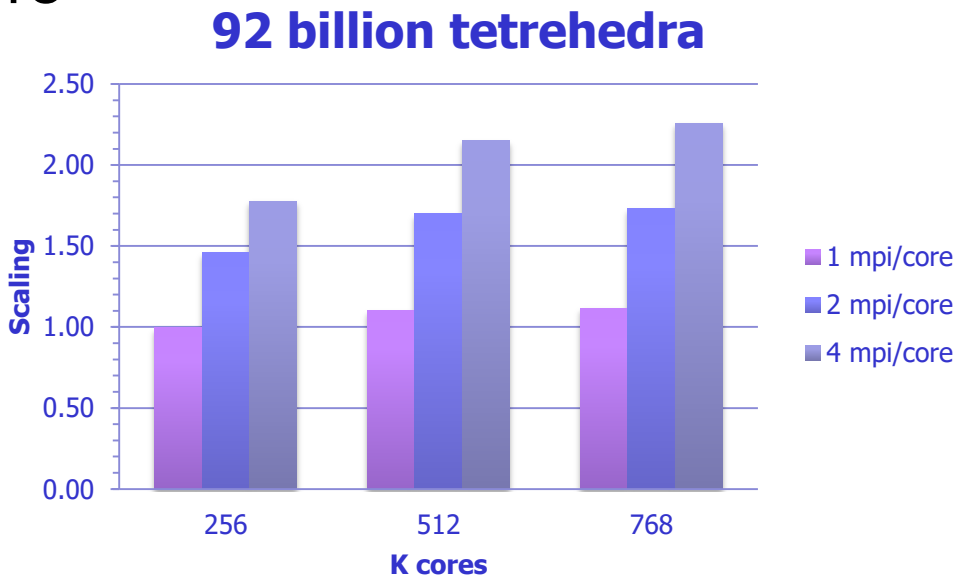
    (ksp/examples/ex35.cxx and ksp/examples/ex36.cxx)
  - ★ Multi-component time-dependent Brusselator reaction-diffusion PDE FEM solver in 1-d. (ts/examples/ex35.cxx)
  - ★ 3-D Nonlinear Laplacian solver (snes/examples/ex36.cxx)
  - ★ 2-D Generalized Finite Difference poisson solver with GMG (ts/examples/GFD/ex2.cxx).

Massively Parallel Unstructured Mesh Solver (PHASTA)

Implicit, Adaptive Grid CFD

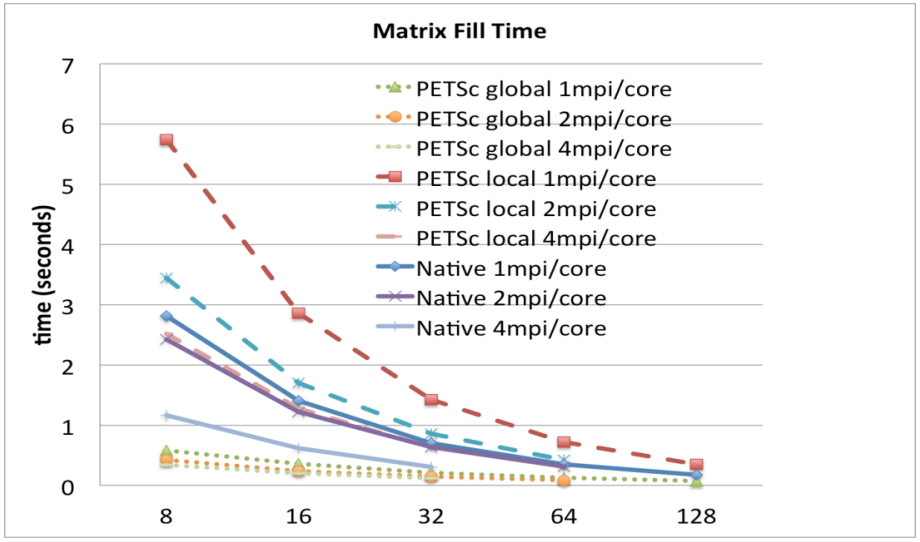
- Extreme Scale Applications:
  - Aerodynamics flow control
  - Multiphase flow
- Full Machine Strong scaling
  - Variable MPI processes/core
  - 92 Billion tetrahedra
  - 262144 to 3,145,728 parts
  - 1/core 100% scaling
  - 2/core 146-155% scaling
  - 4/core 178-226% scaling

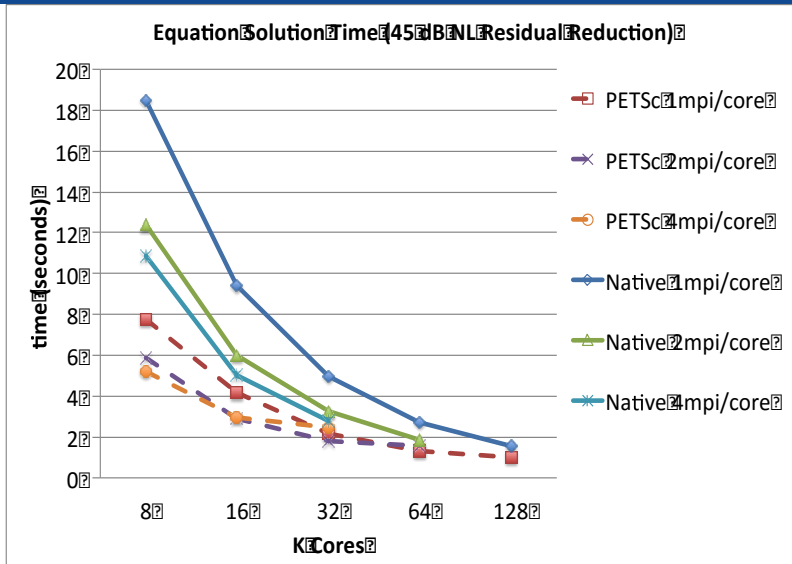


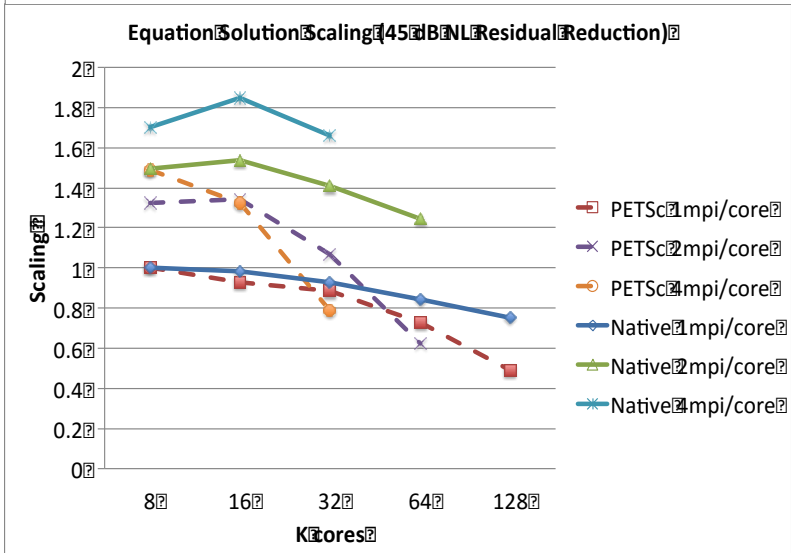


PHASTA/PETSc Coupling

- PETSc functions assemble LHS and RHS including matrix assembly
- New cached assembly – dramatically decreased assembly times
- Need to consider adaptive meshes – can we keep the improvements

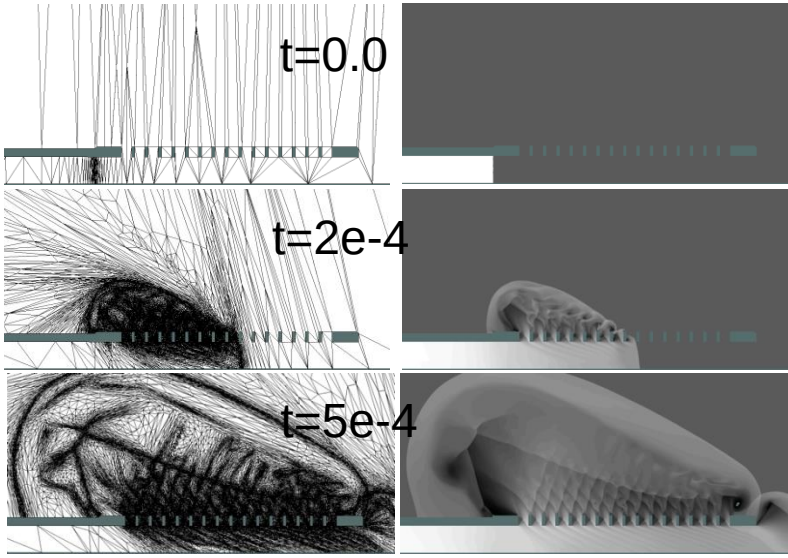


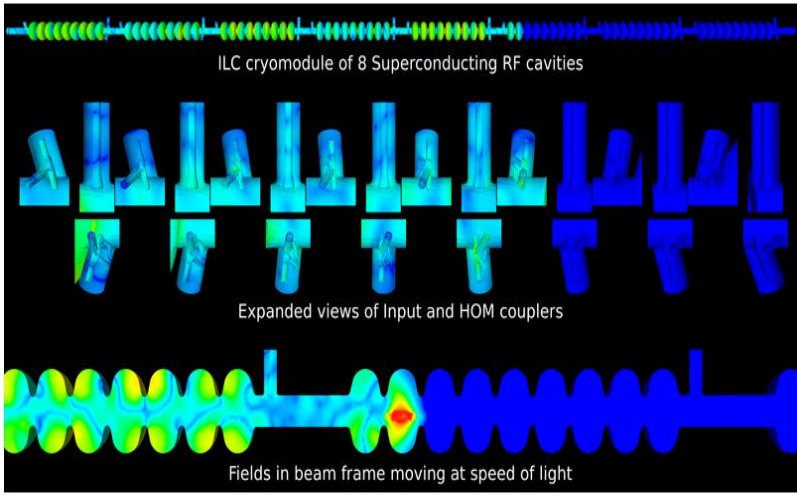




Creation of Parallel Adaptive Loops

- Automation and adaptive methods critical to reliable simulations
- Users want flexibility to apply best in class analysis codes
  - Component-based approach to integrate automated adaptive methods with analysis codes
  - All components operate in parallel, including fast in-memory coupling
- Developing parallel adaptive loops for DOE, DoD and industry using multiple analysis engines

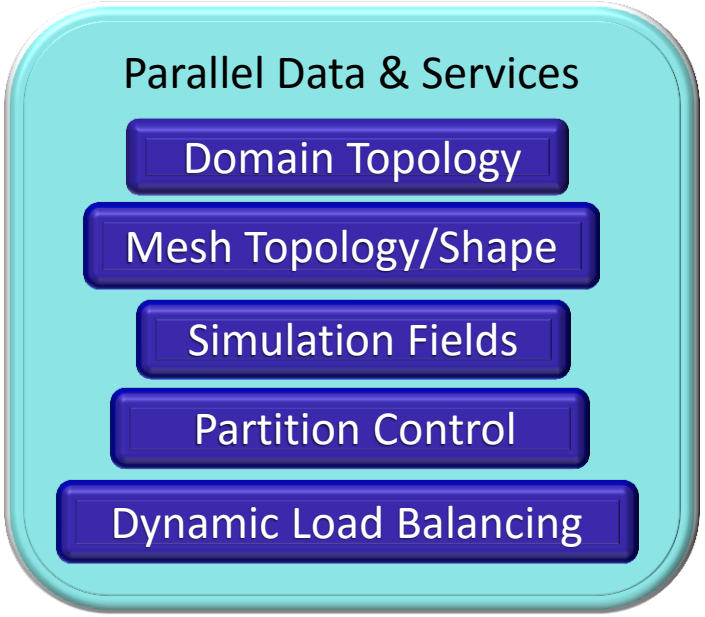




Creation of Parallel Adaptive Loops

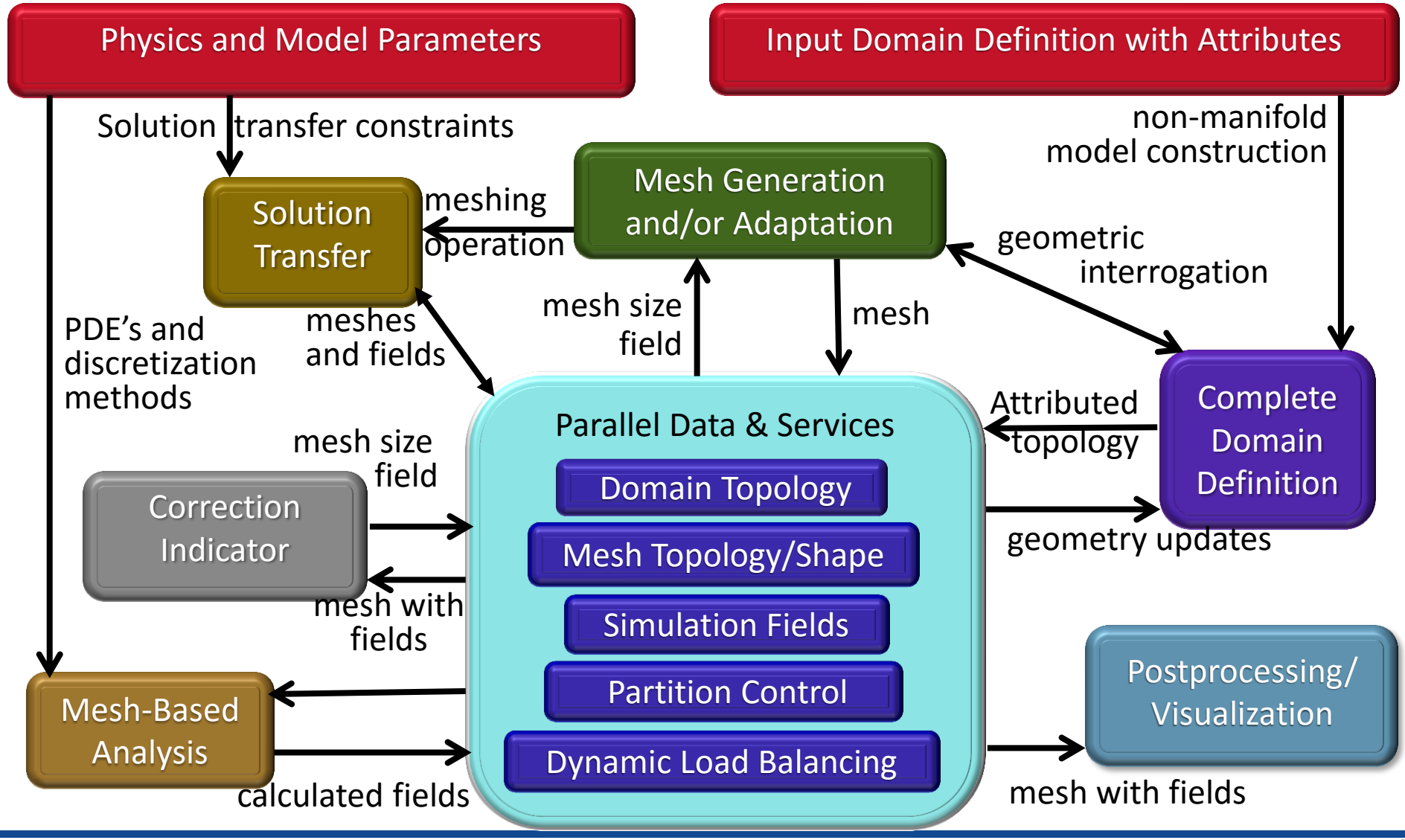
Parallel data and services are the core

- Abstraction of geometric model topology for domain linkage
- Mesh also based on topology – it must be distributed
- Simulation fields distributed over geometric model and mesh entities
- Partition control must coordinate communication and partition updates



- Dynamic load balancing required at multiple steps in the workflow to account for mesh changes and application needs
- Providing parallel data as services with various combinations of FASTMath and other parallel mesh components

Components in Parallel Adaptive Analysis

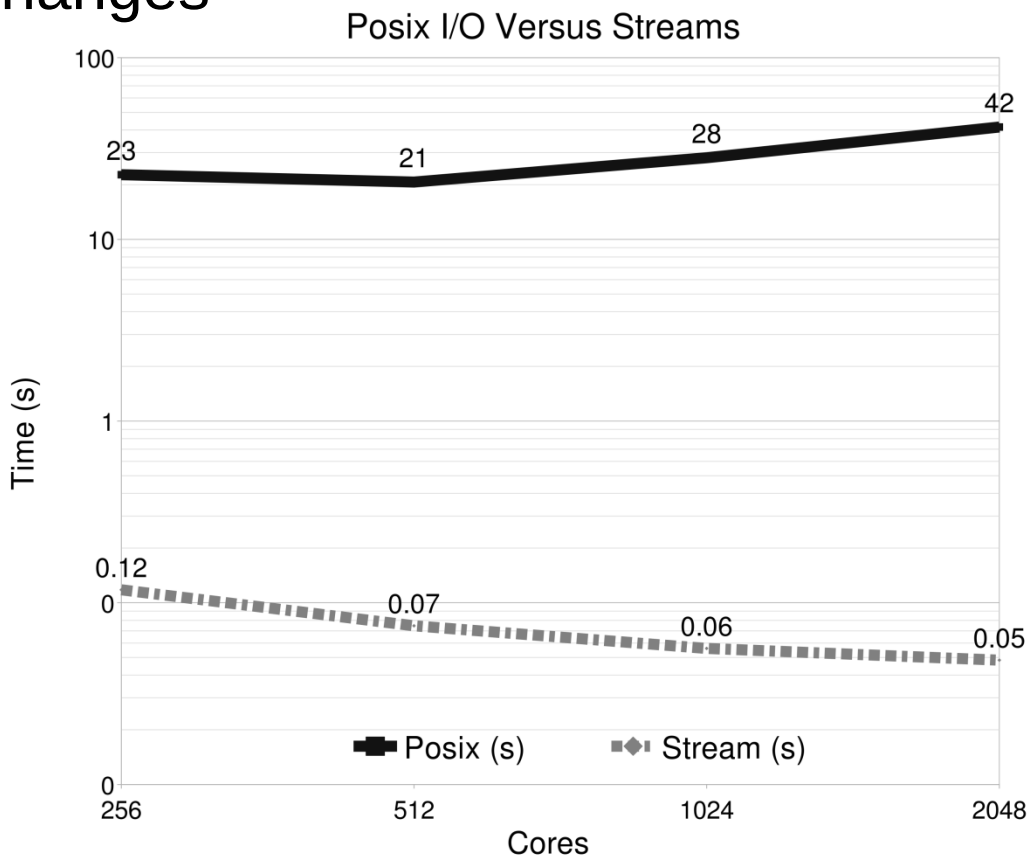


In-Memory Coupling of Simulation Components

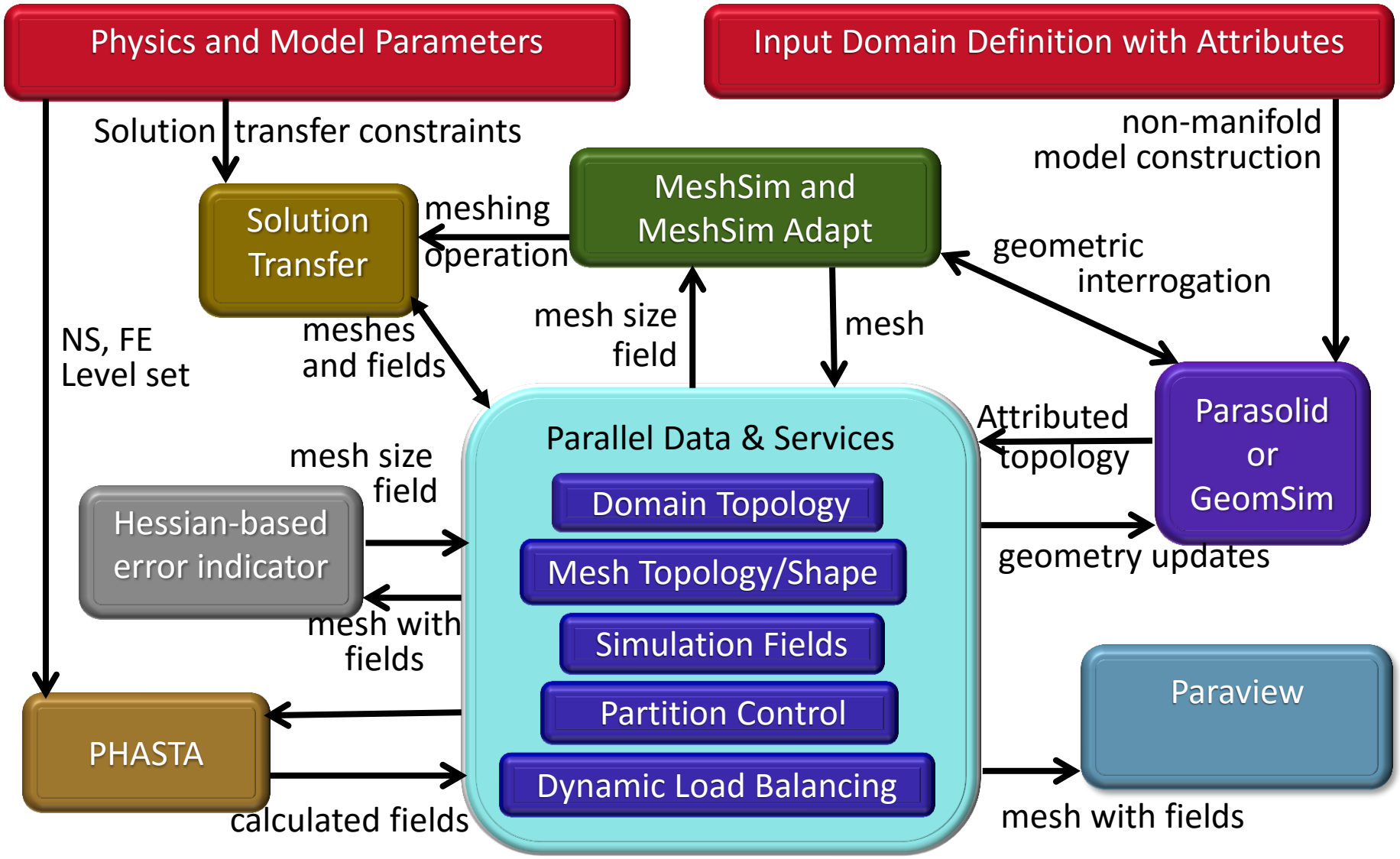
Approaches to avoid significant bottleneck of file based coupling

- Serialized data streams using existing file reading and writing protocol – Minimal code changes
- API based transfer procedures – Flexibility

In-memory has far superior parallel performance



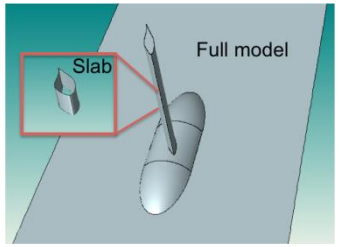
Adaptive Active Flow Control



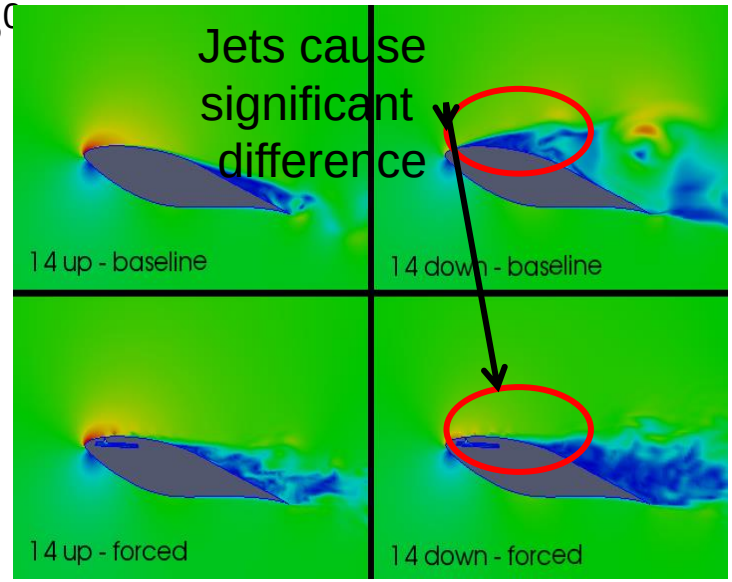
Adaptive Active Flow Control

Dynamic pitch with angle of attack of $14^0 \pm 5.5^0$

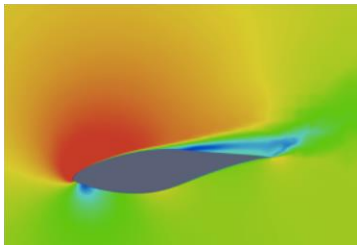
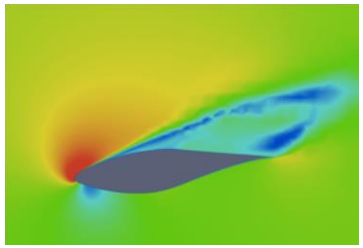
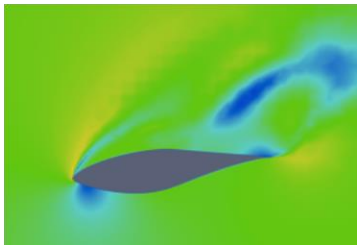
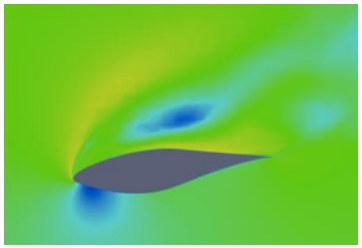
- Slab model – pitch rate of 10Hz
- Baseline (without jets) and forced/controlled (with jets)



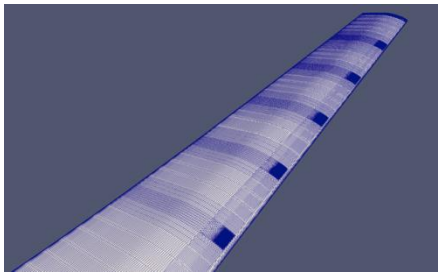
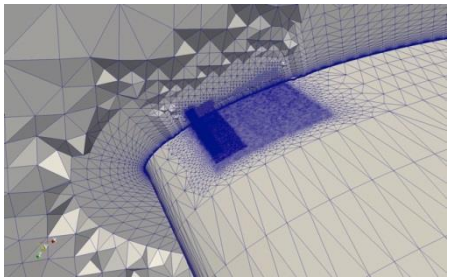
Case\AoA (L/D)	14 Up	14 Down
Baseline	10.2236	3.2286
Forced	10.6265	**9.9921**



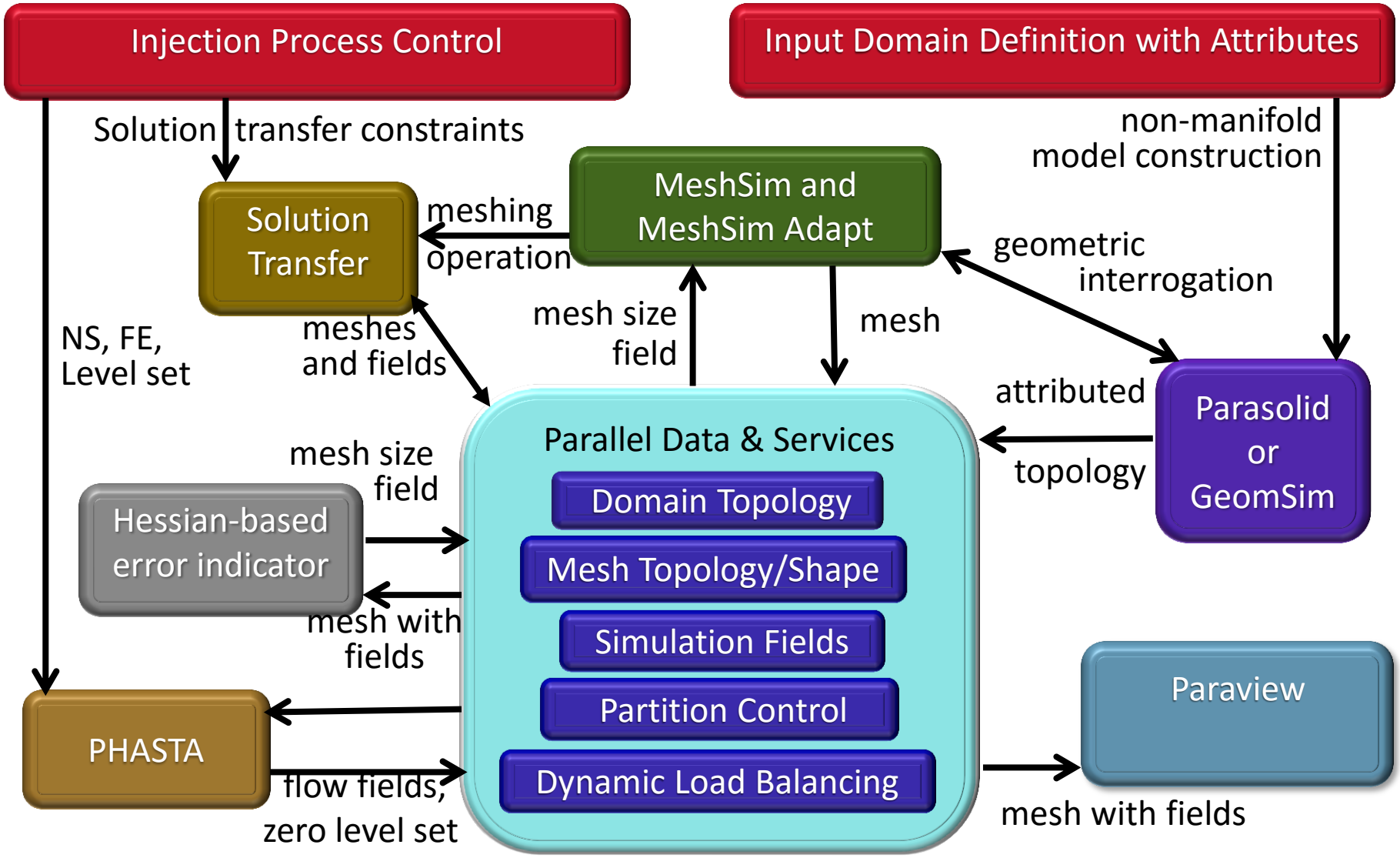
15 m/s

Leading-edge synthetic jets: 5 along span

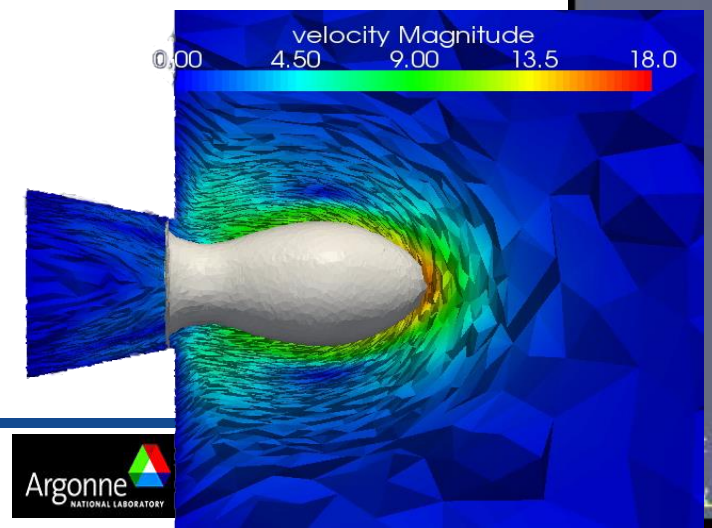



Adaptive Two-Phases Flow

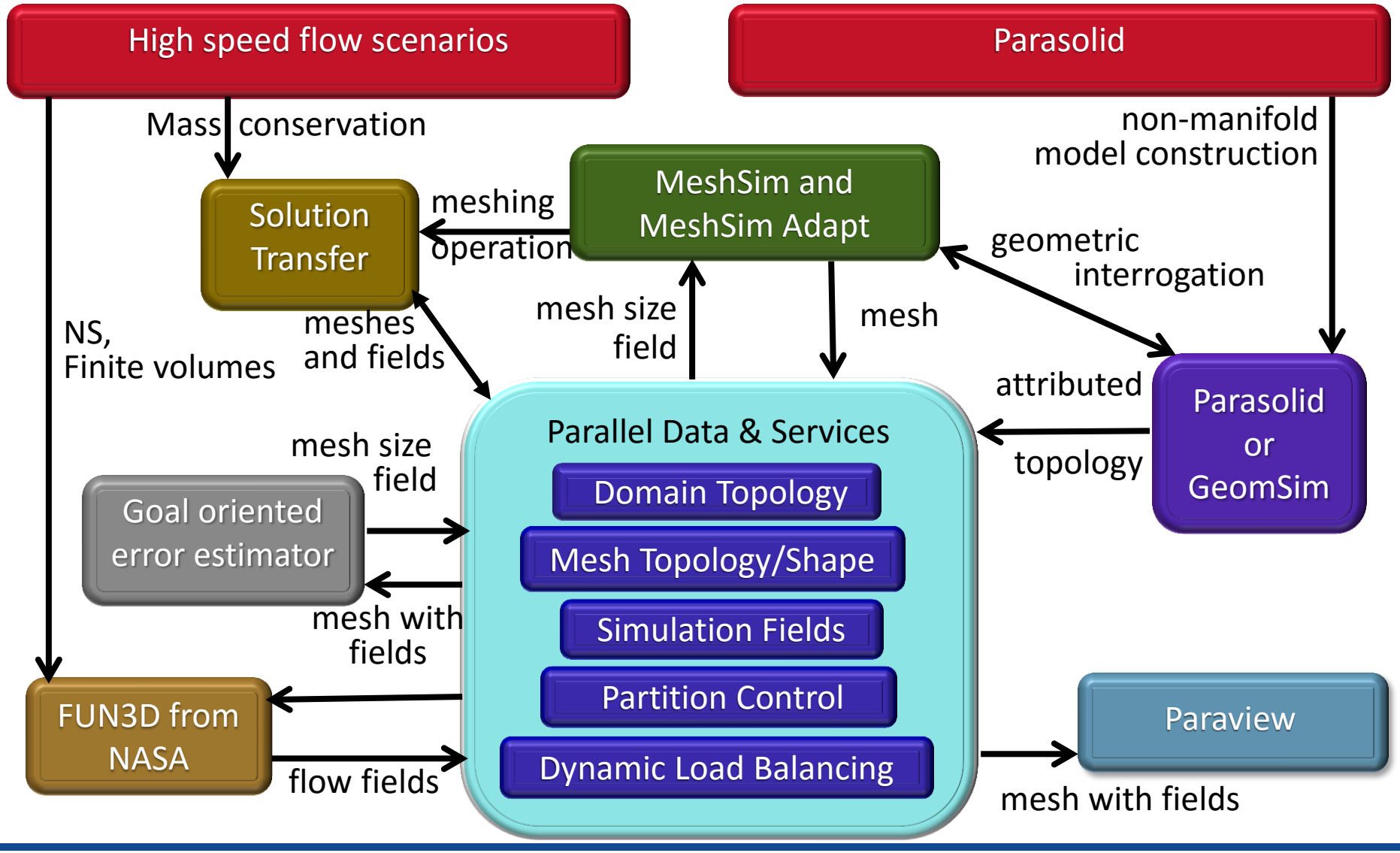


Adaptive Two-Phases Flow

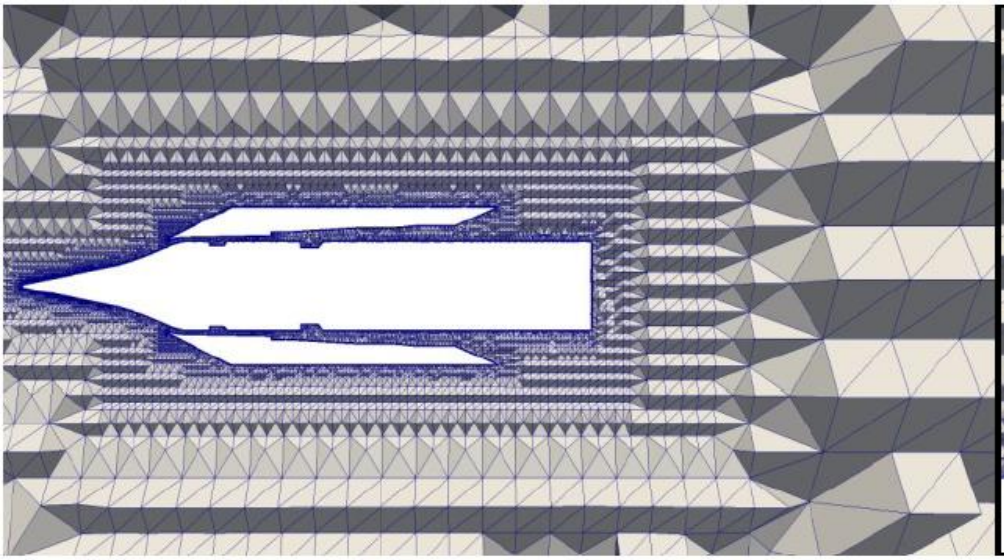
- Two-phase modeling using level-sets coupled to structural activation
- Adaptive mesh control – reduces mesh required from 20 million elements to 1 million elements

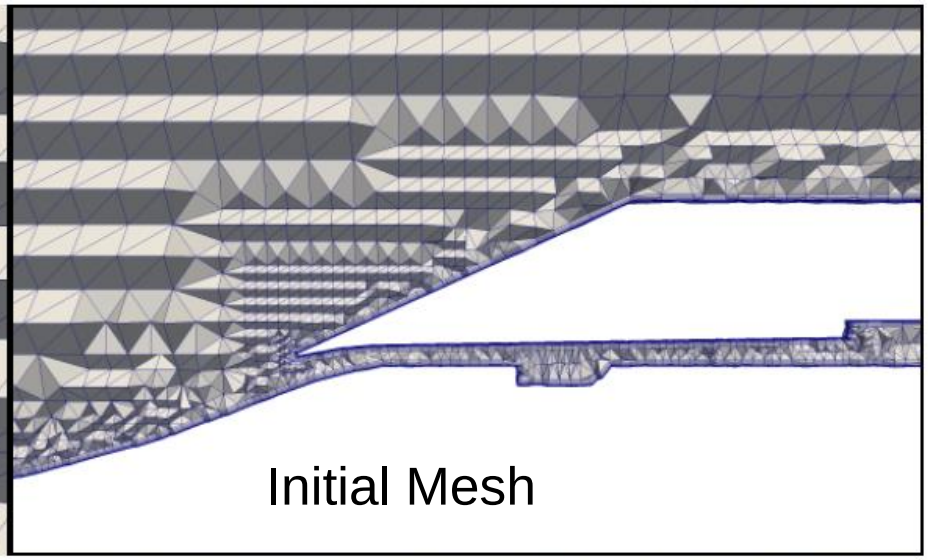


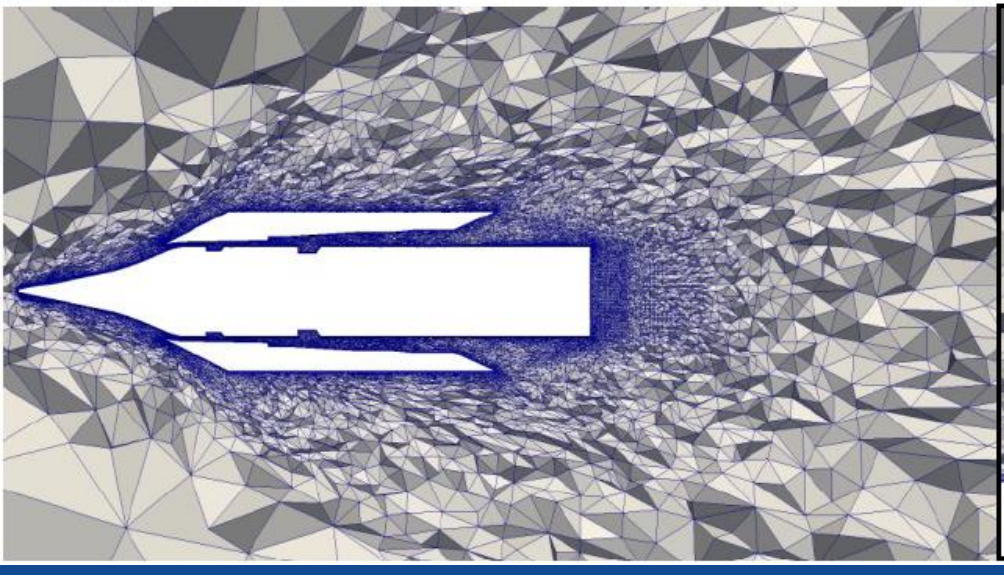
Aerodynamics Simulations

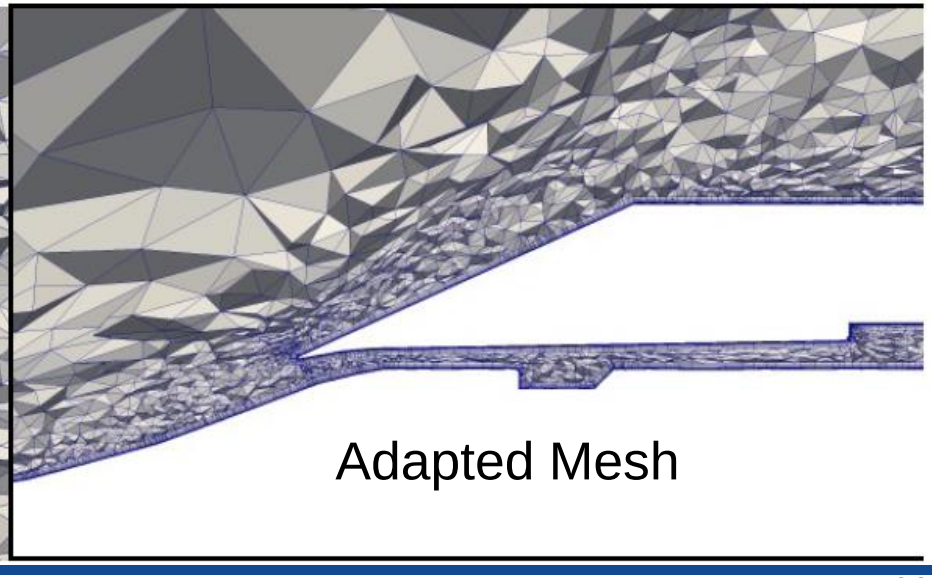


Application Result - Scramjet Engine

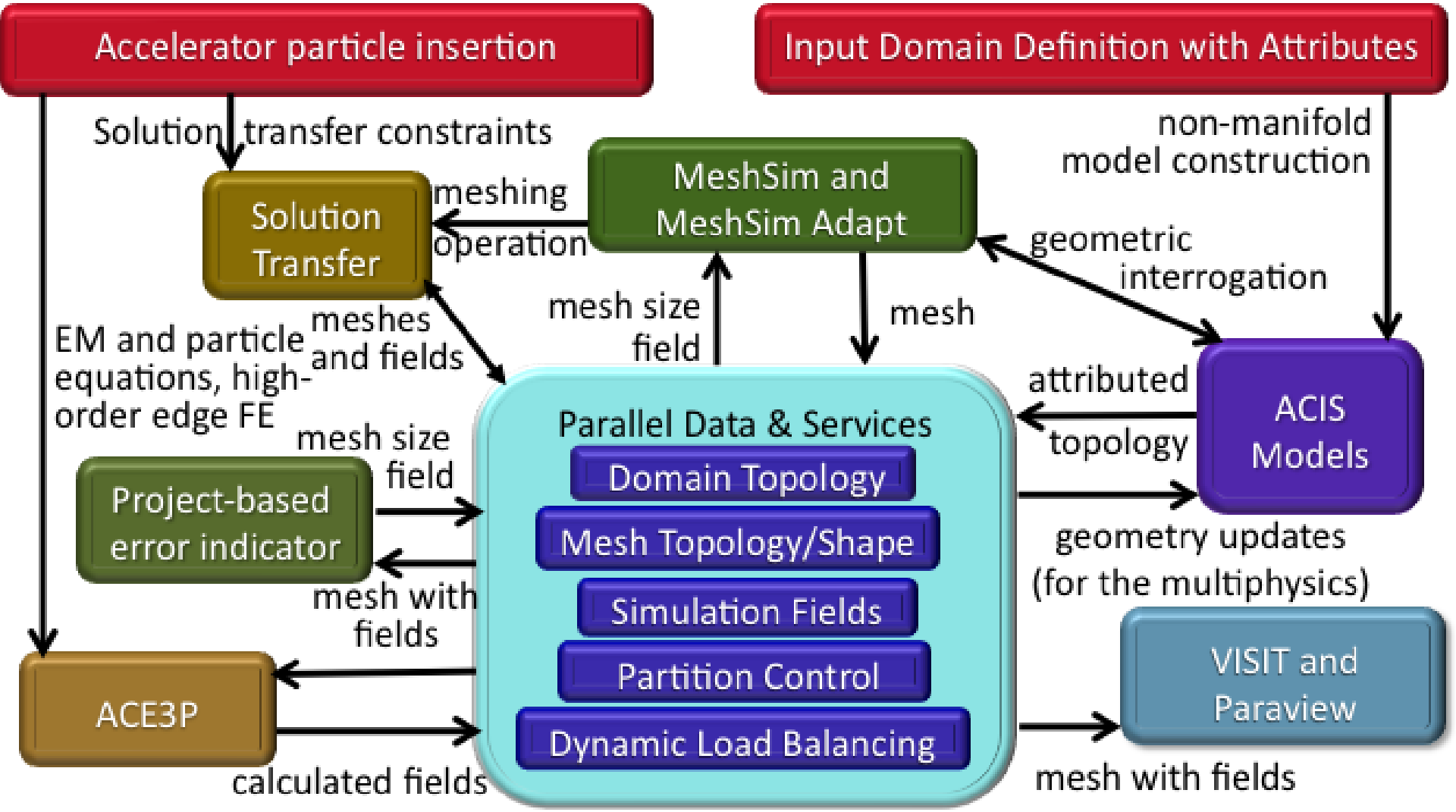






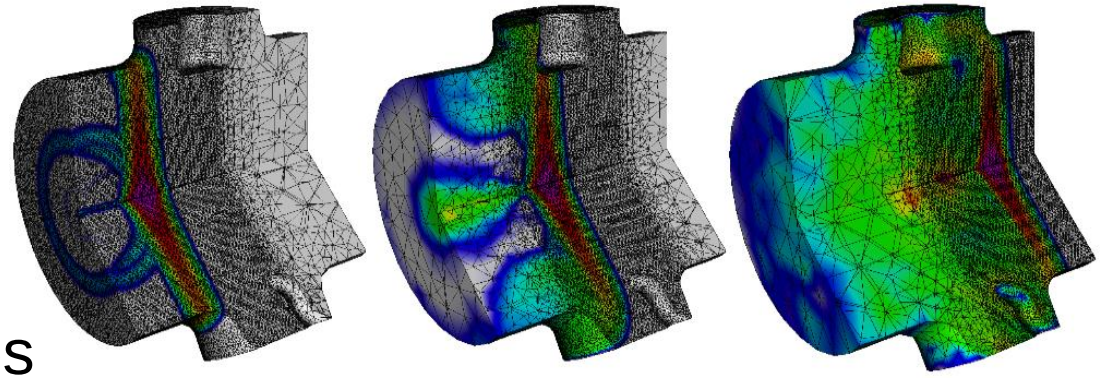


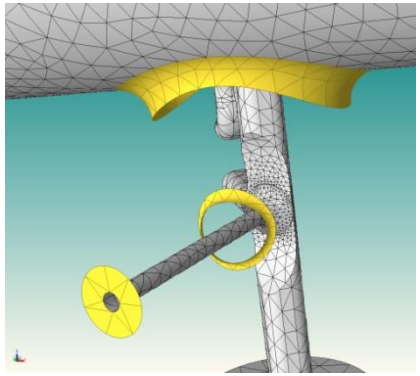
Electromagnetics Analysis

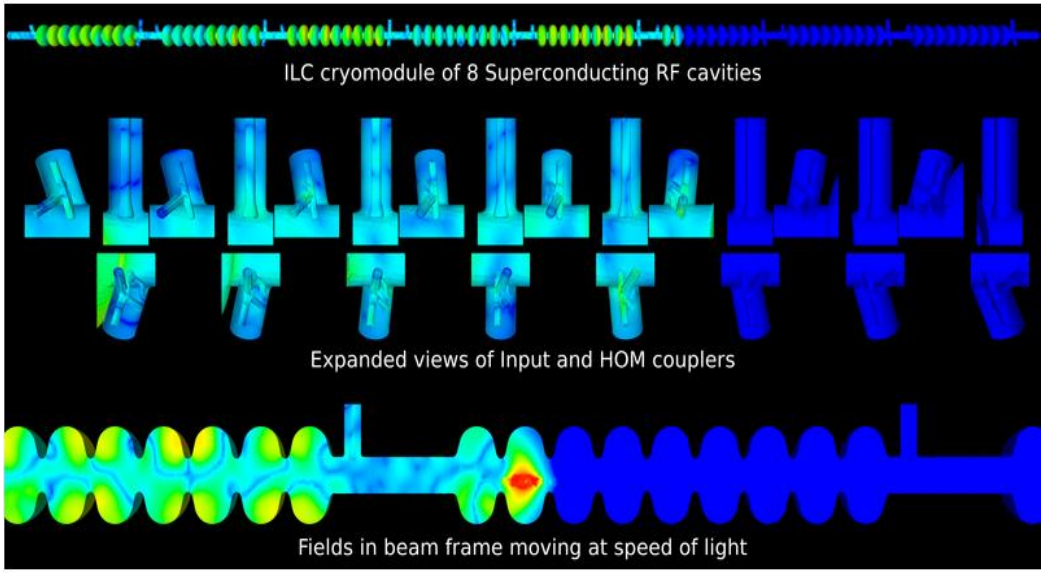


High-Order EM Coupled with PIC

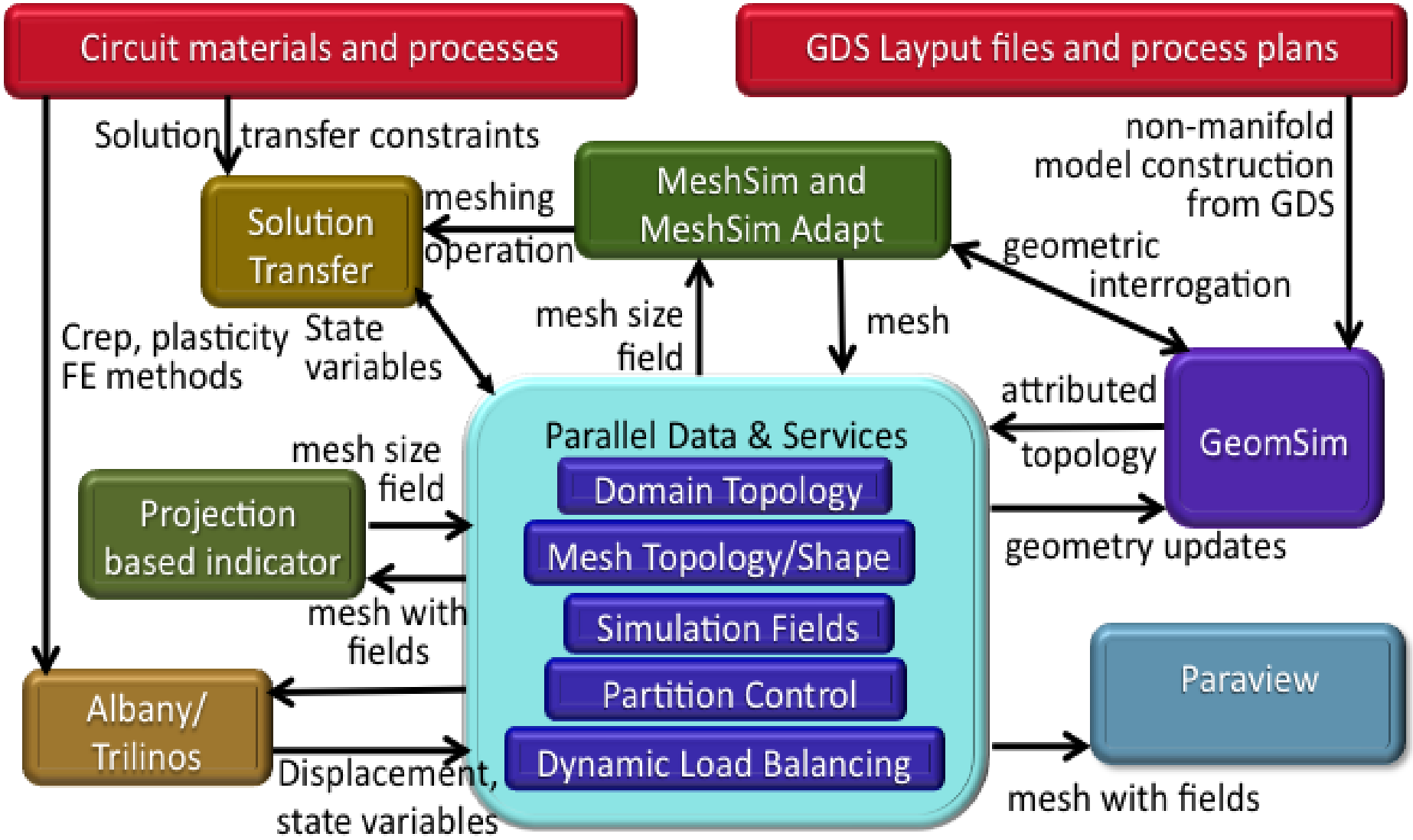
- Adaptation based on
  - Tracking particles
  - Discretization errors
- Full accelerator models
  - Approaching 100 cavities
  - Substantial internal structure
  - Meshes with several hundred million high-order curved elements







Structural Analysis for Integrated Circuits

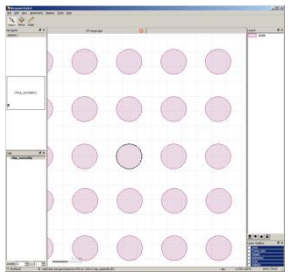


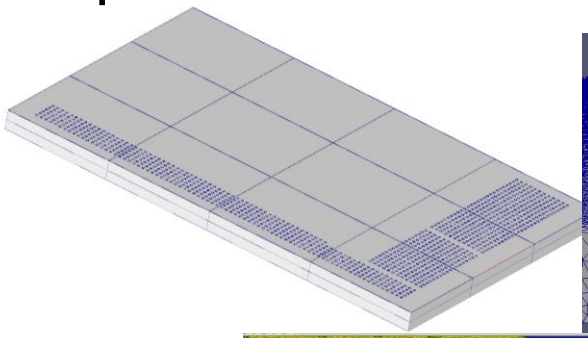
Structural Analysis for Integrated Circuits

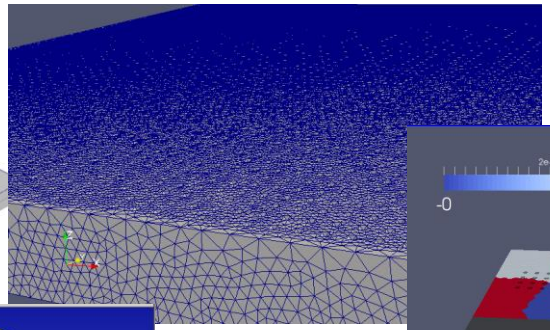
Must construct 3-D non-manifold solid from input geometry

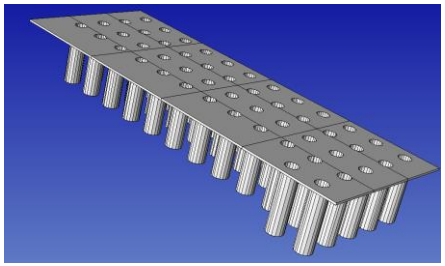
- Input domain defined in terms of 2-D layouts (gdsII/OASIS)
- Third dimension based on process knowledge
- A component has been developed to construct the model

Adaptive loop constructed for thermally loaded case including thin liner

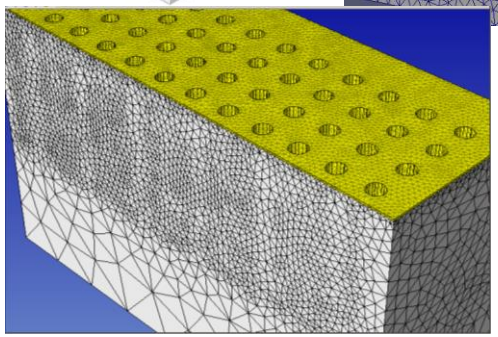


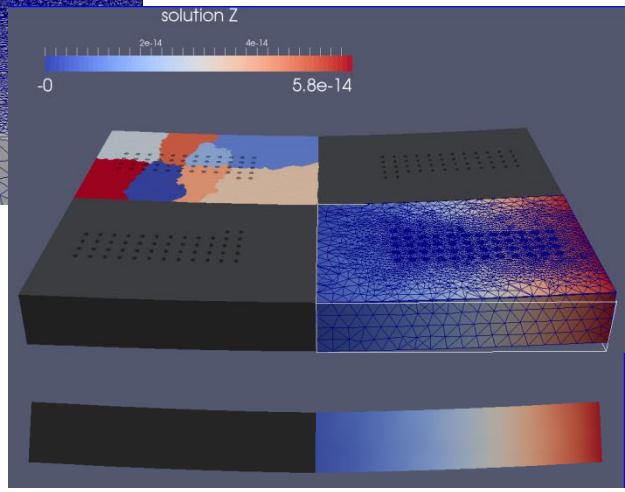






Model of liner film only

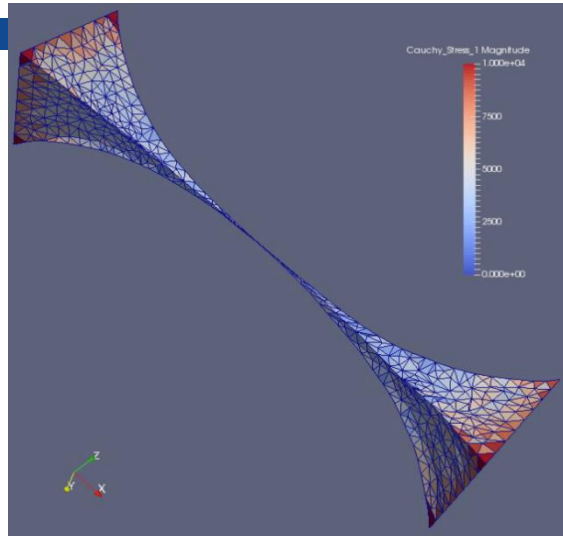


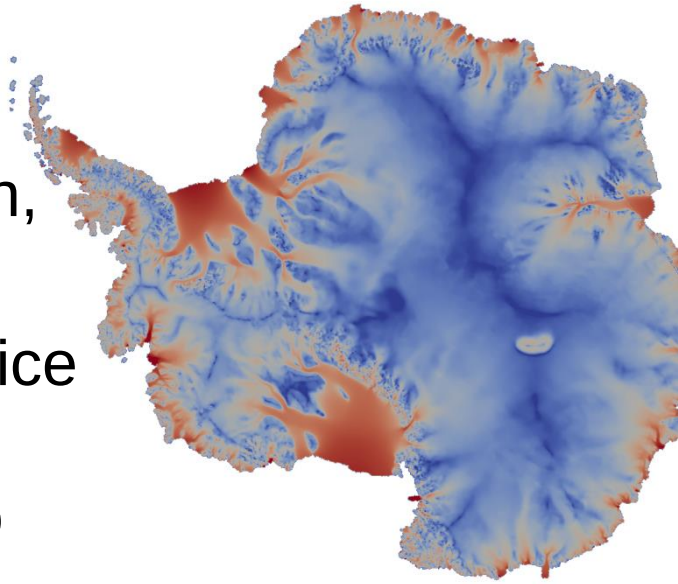


Adaptive Multiphysics Applications

Combination of the FASTMath unstructured mesh technologies with agile multiphysics components and modules to develop:

- ***Albany***: a finite element general implicit multiphysics application for large deformation mechanics, quantum electronics design, CFD, ice sheet and atmosphere modeling, additive manufacturing and topology optimization, multiscale analysis, and more
- ***FELIX***: Albany application for modeling ice sheet dynamics. Features unstructured adaptive meshes, inversion capability to estimate parameters and UQ.

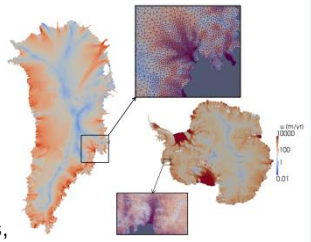




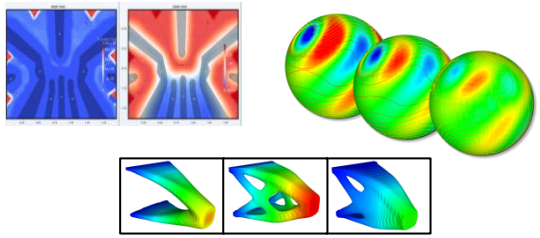
Albany Ecosystem

Application Impact: Ice Sheets

- Surface flow velocities for Greenland and Antarctic Ice Sheets
- Demonstrates nonlinear solves, linear solves, UQ, adaptivity, and performance portability
- Employs automatic differentiation, discretizations, partitioning, mesh database

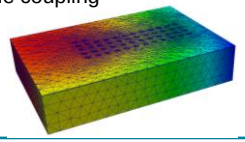


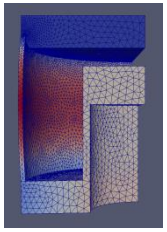
Additional Application Impact



Application Impact: Computational Mechanics

- Largest implicit problem solved in Albany to date: 1.7B degrees of freedom
- Initial capabilities for Schwarz multiscale coupling

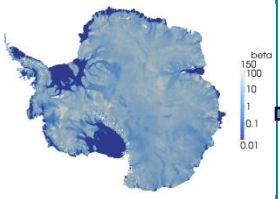




Nonlinear Solvers and Inversion

- Homotopy and Anderson Acceleration in Trilinos::NOX
- The robustness of nonlinear solvers are critical when an application is to be called as a sub-component within a larger application code.
- Uses Automatic Differentiation, Preconditioning, Optimization algorithms from Trilinos

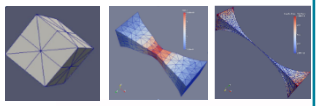
$$\frac{dg}{dp} = \frac{\partial g}{\partial x}^T \frac{\partial f}{\partial x}^{-1} \frac{\partial f}{\partial p} + \frac{\partial g}{\partial p}$$

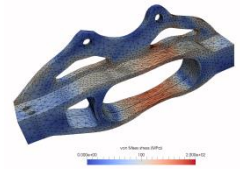


Application

Mesh Adaptivity

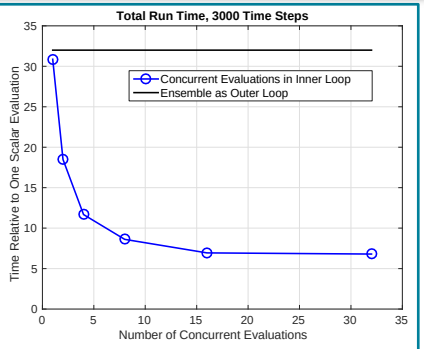
- Mesh adaptation can be essential for efficiency and robustness
- Cube geometry subjected to large deformation (elasticity and J2 plasticity results shown)





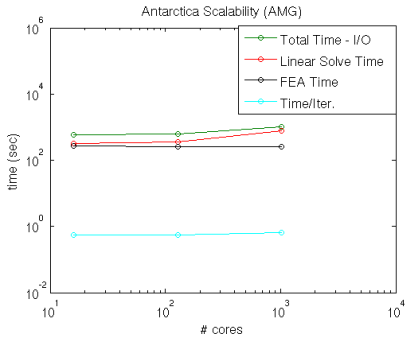
Embedded UQ

- New *Ensemble* data type in Sacado package
- Vectorization of kernels over ensembles
- Contiguous memory access in arrays



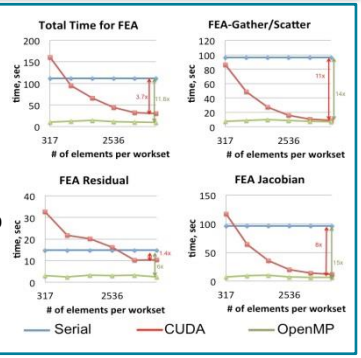
Scalable Linear Algebra

- Scalability of simulations requires effective preconditioning
- Multi-level solves are essential for the largest problems

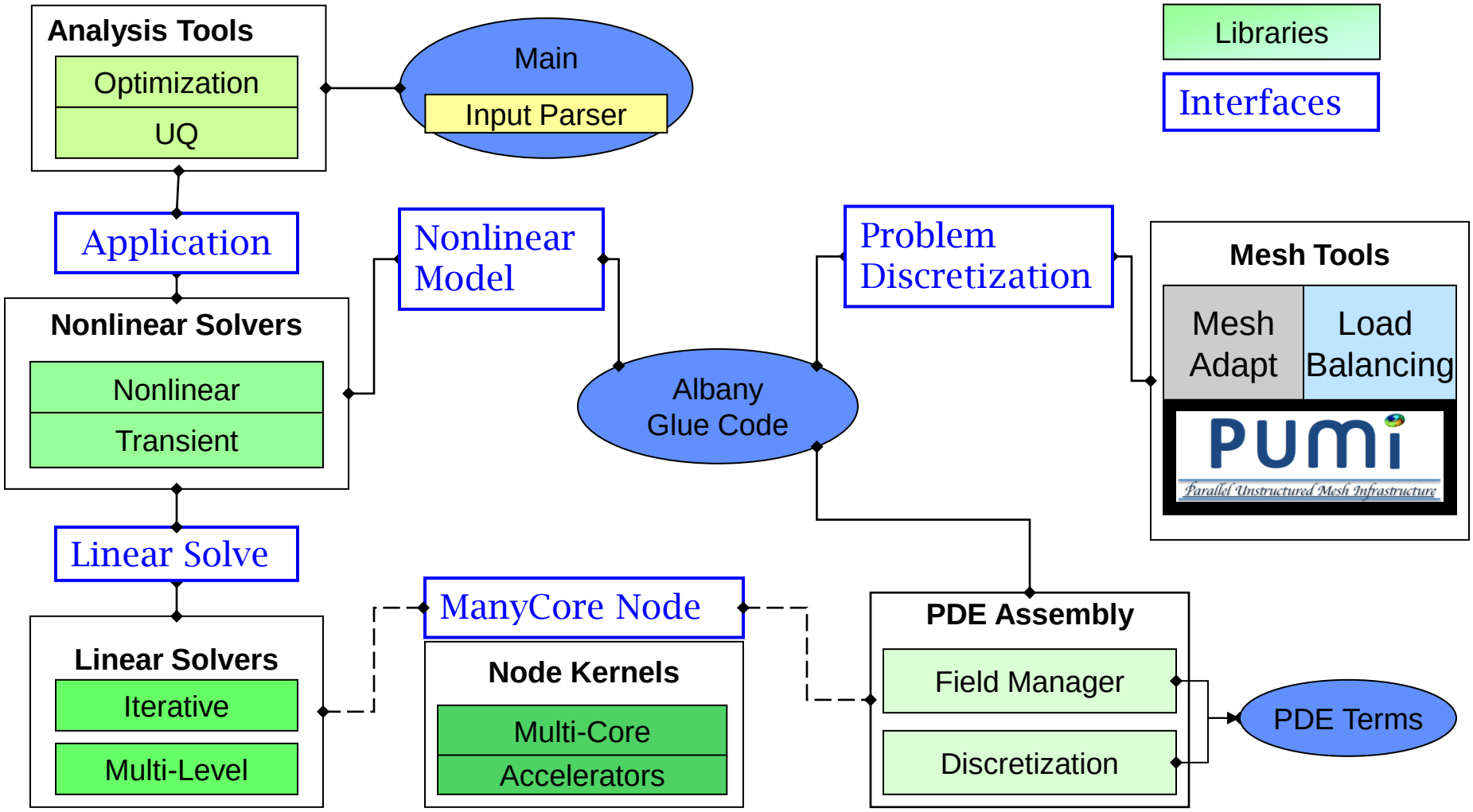


Performance Portability

- The Kokkos programming mode supports performance portability of kernels.
- Kokkos' abstraction layer allows code to be tailored for specific devices



Albany – Agile Component Architecture



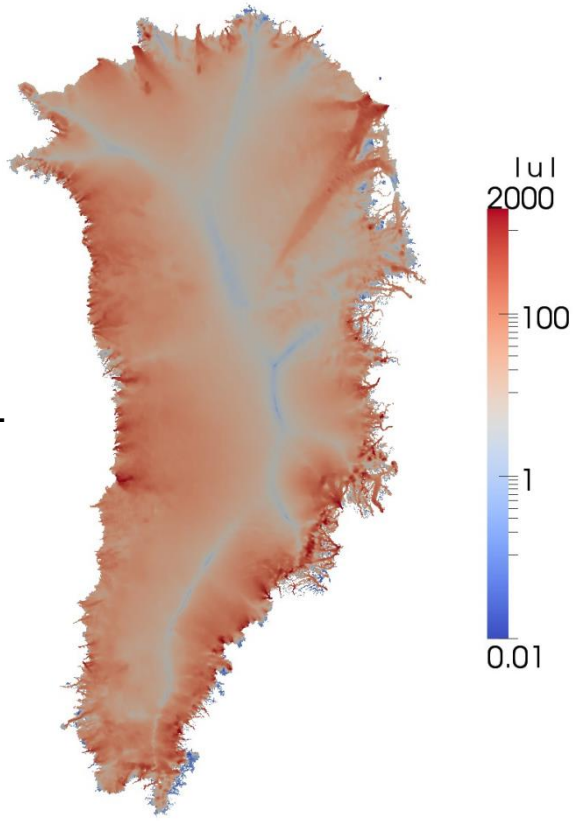
Integrated Technologies Highlight: PISCEES ASCR/BER SciDAC: Albany/FELIX

ASCR/BER

Develop and **support** a robust and scalable land ice solver based on the "First-Order" (FO) Stokes physics

- Uses dozens of libraries from Trilinos
- Robust nonlinear solves (NOX + LOCA)
- Scalable Multi-level linear solves (ML)
- Large scale, adjoint-based, inversion capability using ROL
- Jacobians and adjoint computed using AD (SACADO)
- Embedded in-memory mesh adaptation (PAALS)
- Part of next-generation DOE climate model (ACME)

❖ Linked to Dakota
for UQ and Bayesian Inversion
(KLE → PCE → AS → MCMC)

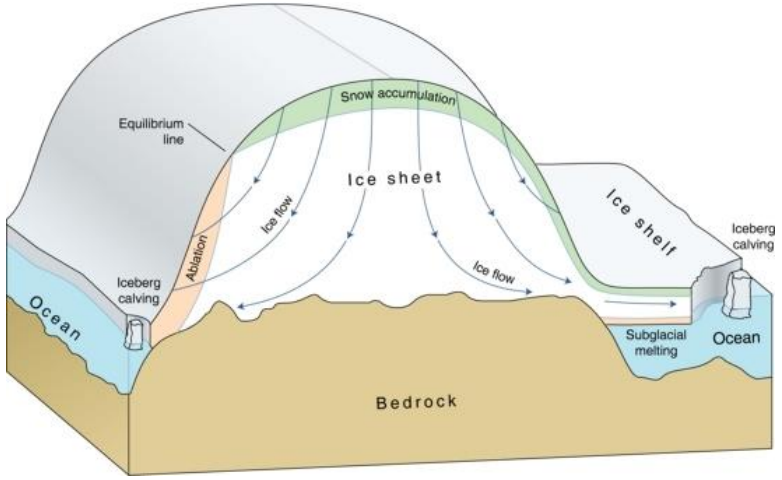


Greenland Ice Sheet
Surface Velocities

Albany/FELIX Ice Sheet Modeling

ASCR/BER

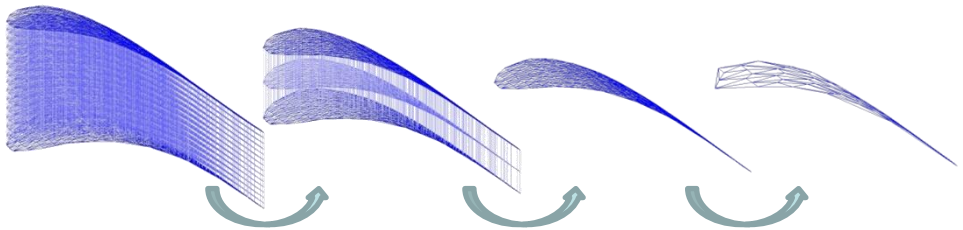
- Ice behaves like a very viscous shear-thinning fluid (similar to lava flow) and can be modeled with a nonlinear Stokes equation.
- Greenland and Antarctica ice sheets have a shallow geometry (thickness up to 3km, horizontal extensions of thousands of km).
- Exploiting their shallow nature, ice sheet meshes are obtained by extruding an unstructured 2d grid.
- Adaptation is performed on the 2d grid, based on the gradient of the surface velocity.

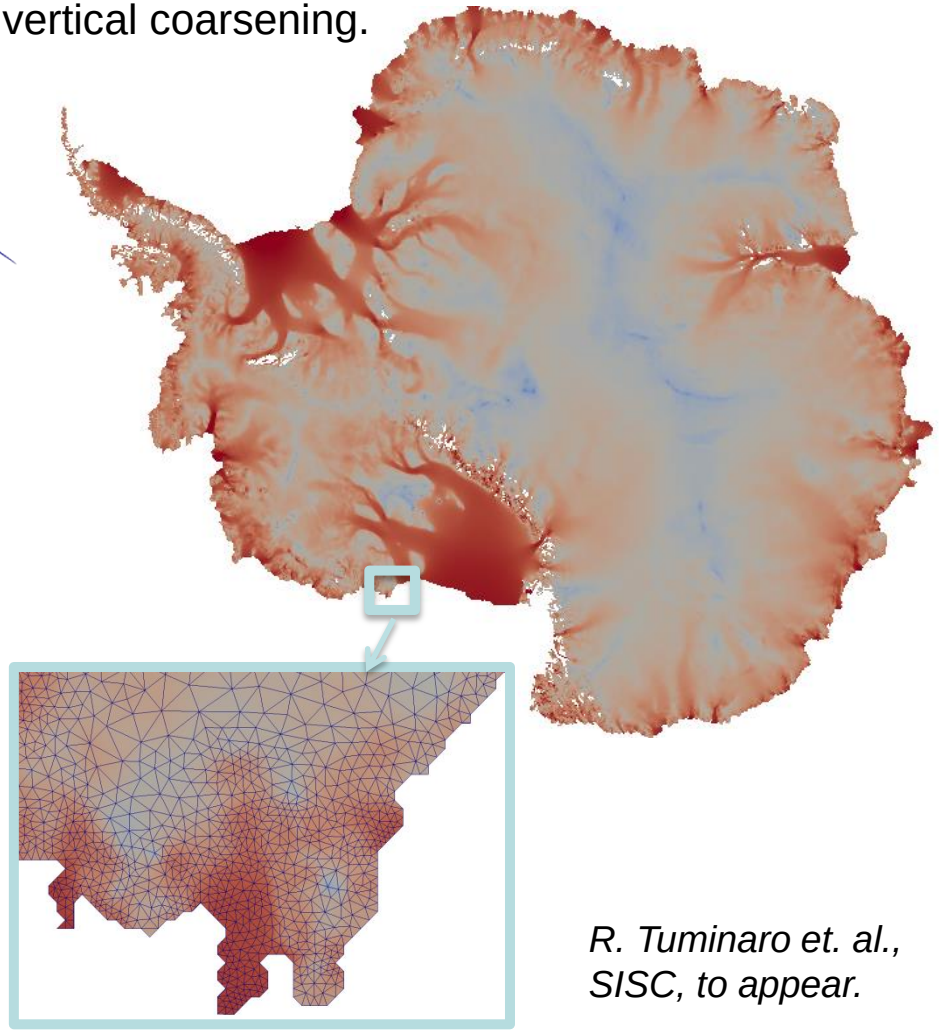



Getz ice shelf, by NASA/Dick Ewers

Albany/FELIX Parallel Scalability

Scalability achieved using AMG preconditioner that leverages extruded structure of the meshes, by aggressive vertical coarsening.





< 2x solution time increase
(2.5 million dofs → 1.1 billion dofs)

# cores	# dofs	avg its. per solve	total lin. seconds
16	≈ 2.5 m.	9.2	220.
128	≈ 18.5 m.	11.6	245.
1024	≈ 141.5 m.	14.6	294.
8192	≈ 1.1 T	20.2	378.

R. Tuminaro et. al., SISC, to appear.

Albany/FELIX Inversion Capability

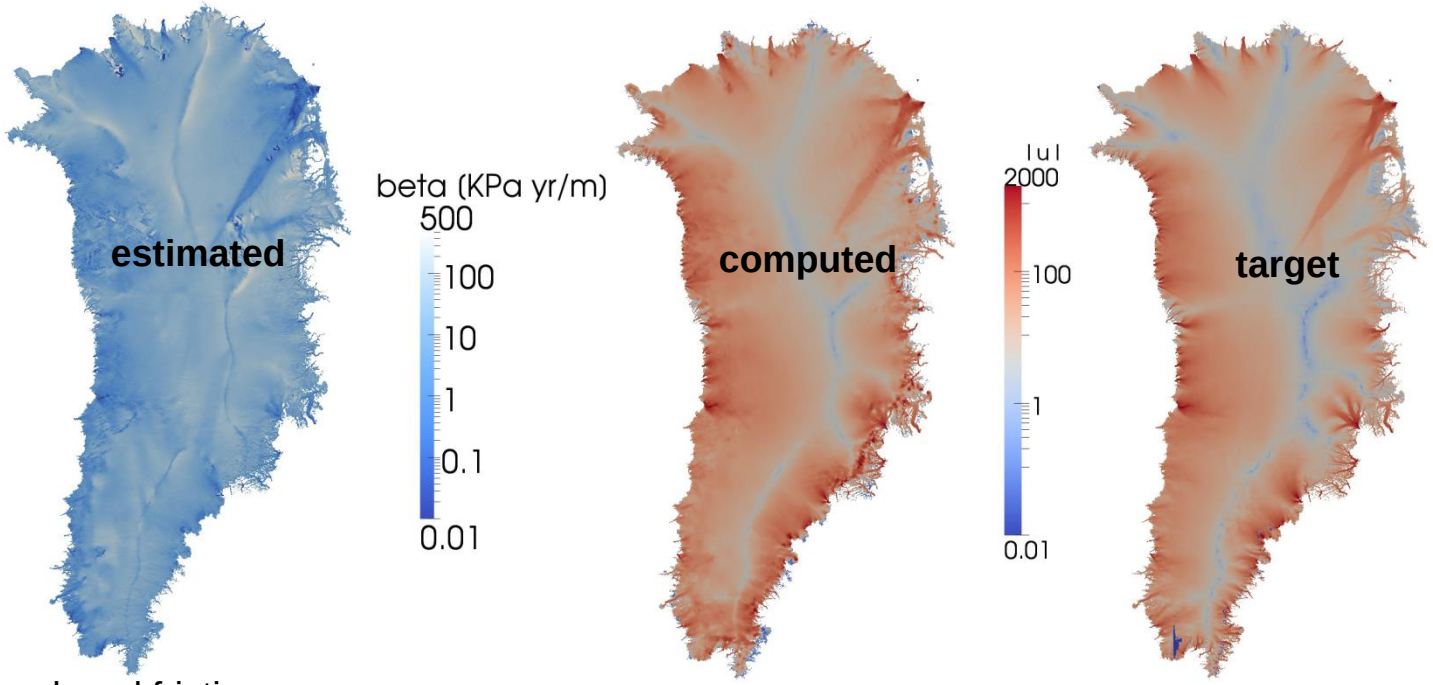
Basal friction (β) at the ice bed is unknown and we need to estimate it.
PDE-constrained optimization approach: find β that minimize the mismatch with observations

$$\mathcal{J}(\mathbf{u}(\beta),\ \beta) = \int_\Sigma \frac{1}{\sigma_u^2}|\mathbf{u} - \mathbf{u}^{obs}|^2\, ds + \ \alpha \int_\Sigma |\nabla\beta|^2\ ds$$

(velocity mismatch + regularization)

Optimization Algorithm:
L-BFGS (ROL)

DOF: 35 m.
Params: 1.6 m.



basal friction

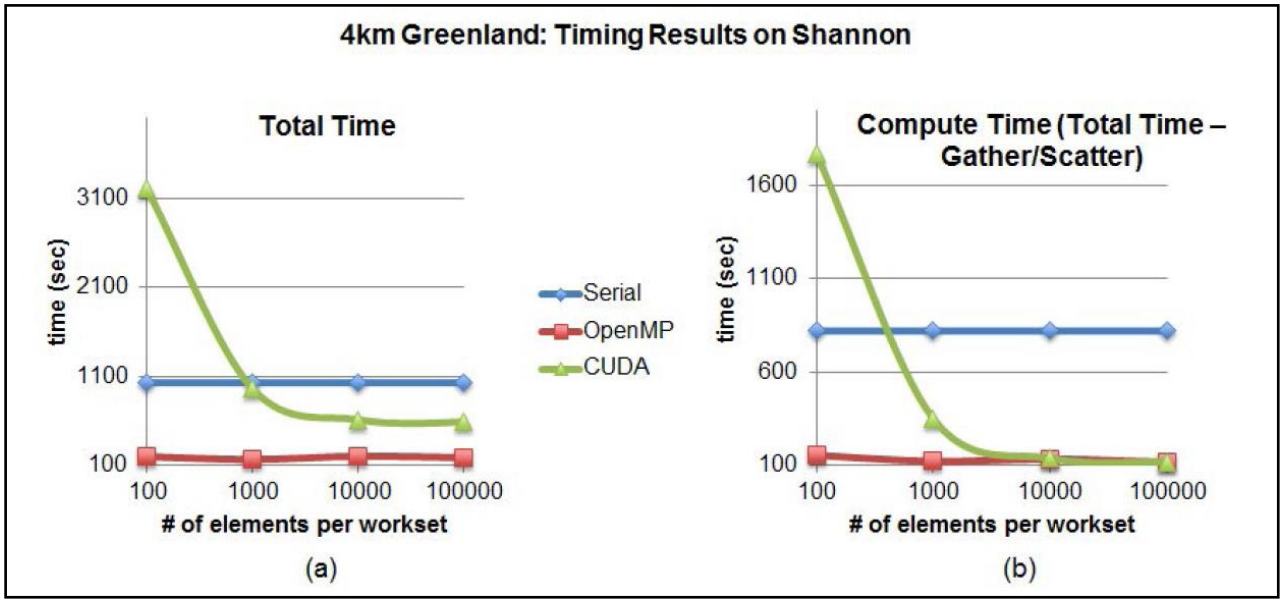
surface ice velocity

M. Perego et. al., JGR, 2014.

Albany/FELIX Parallel Portability

- Albany finite element assembly can be run with OpenMP and CUDA, using the Kokkos library (C++ library for manycore performance portability).

- Results below show on-node performance (strong scalability)
  - Serial: 2 MPI processes per node
  - OpenMP: 16 Kokkos OpenMP threads per node
  - GPU : 1 NVIDIA K80 GPU per node



Shannon:
32 nodes:
Two 8-core Sandy Bridge Xeon E5-2670 @ 2.6GHz (HT deactivated) per node,
2x NVIDIA K20x/K40 per node

Note: Gather and Scatter require copying data between host and GPU.

Demeschko I. et al., 2016, submitted.

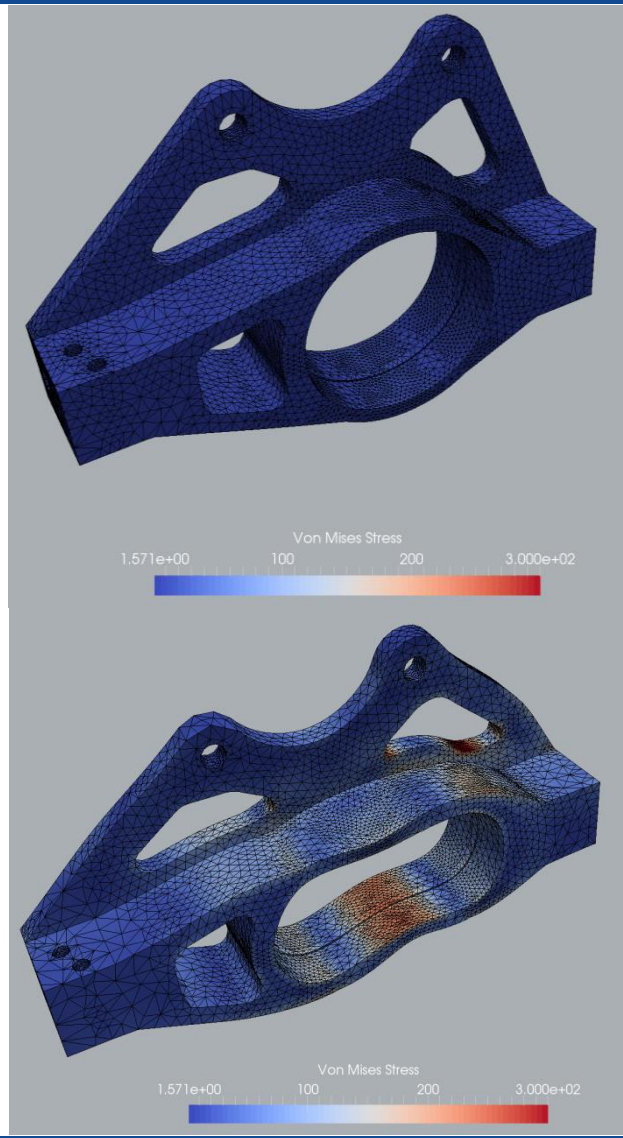
Modeling Large Deformation Structural Failure

Adaptive simulations of finite deformation plasticity with Albany

- Projects include modeling large deformation and weld failures

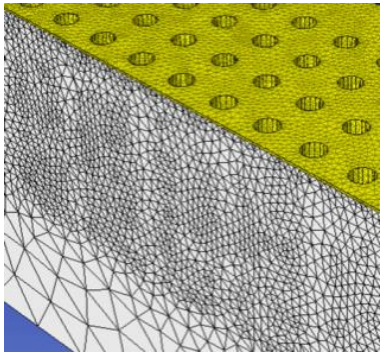
Efforts on adaptive loops that supports

- Solution accuracy via error estimation
  - Error estimation library
- Element shape controlled each load step
- Accurate state variable transfer
- Predictive load balancing (ParMA, Zoltan) at each adaptive stage
- Adding adjoint capabilities for goal oriented error estimation

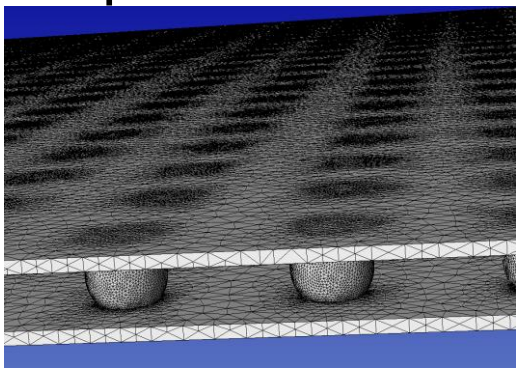


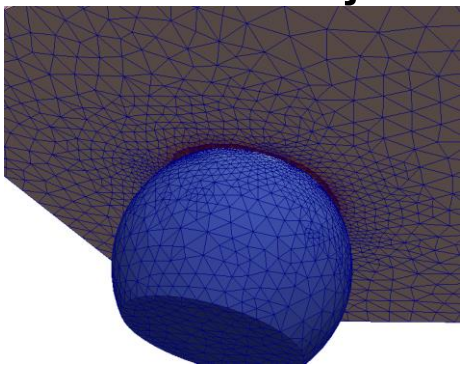
Mechanical Failures in Integrated Circuits

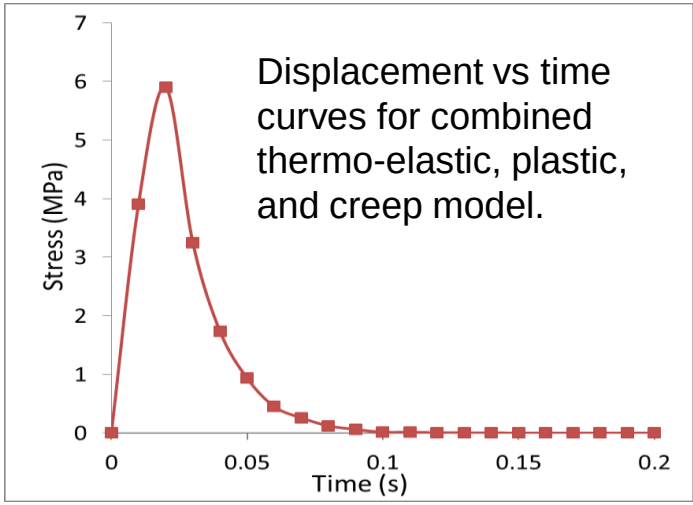
Microelectronics processing is very exacting and mechanical responses impact reliability and manufacturability



- Multi-layer nature of chips interacts with thermal cycles, creep, and intrinsic stress of films
- Intrinsic stress in film deposited onto surface and into features causes macroscopic deflection of wafer
- Combined thermoelastic, plastic, and creep model constitutive model implemented in ALBANY
- Creep and delamination in solder joints







Displacement vs time curves for combined thermo-elastic, plastic, and creep model.

FASTMATH

FASTMath Unstructured Mesh Hand-On Session

Two tracks:

- **Use SIGMA tools to construct mesh and its discretization for solving 2-D/3-D Poisson PDE**
- **Workflow demonstration using Simmetrix/PUMI/PAALS for parallel adaptive simulations**

FASTMATH

Tutorial Session for Scalable Interfaces for Geometry and Mesh based Applications (SIGMA)

ATPESC 2016

Vijay Mahadevan and Iulian Grindeanu

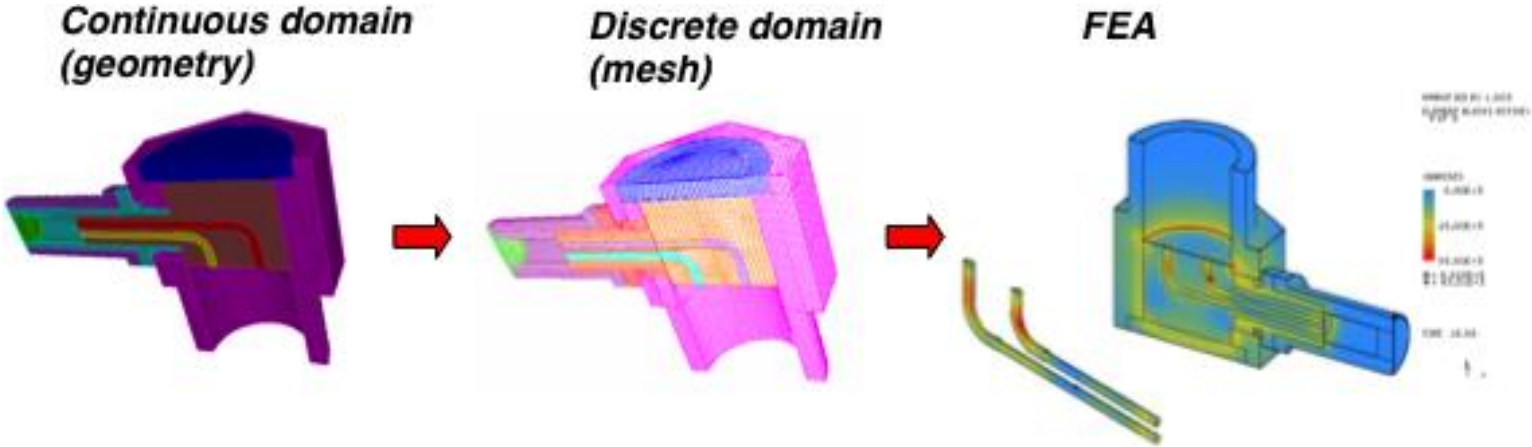
FASTMath SciDAC Institute

Argonne NATIONAL LABORATORY · BERKELEY LAB · Berkeley · Lawrence Livermore National Laboratory · Rensselaer · Sandia National Laboratories · SMU · STONY BROOK UNIVERSITY · UBC THE UNIVERSITY OF BRITISH COLUMBIA · WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

SIGMA: Introduction

Website: http://sigma.mcs.anl.gov

- **Capabilities:** Geometry and Mesh (data) generation/handling infrastructure with flexible solver interfaces.



- **CGM** supports both open-source (OpenCascade) and facet-based (STL) geometry modeling representations.
- **MOAB** provides scalable mesh (data) usage in applications through efficient array-based access; Support parallel I/O, solution transfer, visualization.
- **MeshKit** provides unified meshing interfaces to advanced algorithms and to external packages (Cubit/Netgen).
- **PETSc – MOAB** interface (DMMoab) simplifies efficient discretization and solution of PDE on MOAB unstructured meshes with FD/FEM.

SIGMA Tutorial

Website: http://sigma.mcs.anl.gov

- To understand the usage of SIGMA tools, follow the simple workflow to solve a 3-D Laplacian on an unit cube mesh.
  - ★ Example 1: HelloParMOAB
    - Introduction to some MOAB objects and load mesh in parallel
    - Query the parallel mesh to list the entities of various dimensions (elements, faces, edges, vertices)
  - ★ Example 2: LargeMesh
    - Generate *d*-dimensional parallel meshes with given partition/element information (HEX/TET/QUAD/TRI)
    - Define Tags on entities (vertex or elements)
    - Write to file in parallel with partition
  - ★ Example 3: MBPart
    - Partition mesh with Metis and/or Zoltan

SIGMA Tutorial

Website: http://sigma.mcs.anl.gov

- ★ Example 4: LloydSmoother
  - Improve mesh quality and uniformly refine resolution in parallel
- ★ Example 5: 2-D/3-D DMMoab Poisson Solvers
  - Introduction to some DMMoab concepts
  - Describe and define fields to be solved
  - Associate mesh hierarchy to geometric multigrid preconditioner
  - Setup linear operator and solve in parallel
  - Output mesh/solution and visualize

- Please consult the SIGMA website for help on examples.

  http://sigma.mcs.anl.gov/sigma/atpesc2016
- All other MOAB questions can be directed to

  moab-dev@mcs.anl.gov

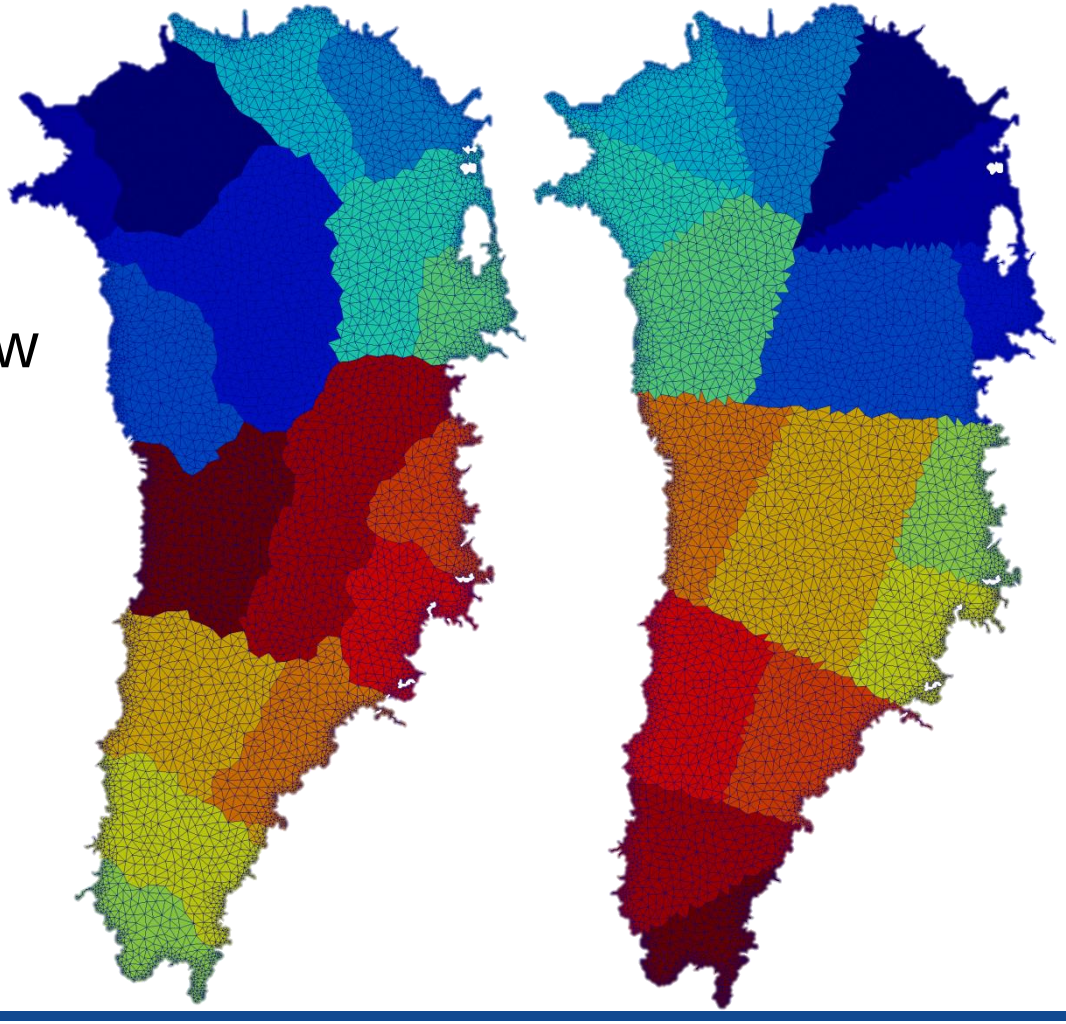
Workflow demonstration using Simmetrix/PUMI/PAALS for parallel adaptive simulations

Presenters: Cameron W. Smith, Mauro Perego and Glen Hansen

FASTMath SciDAC Institute

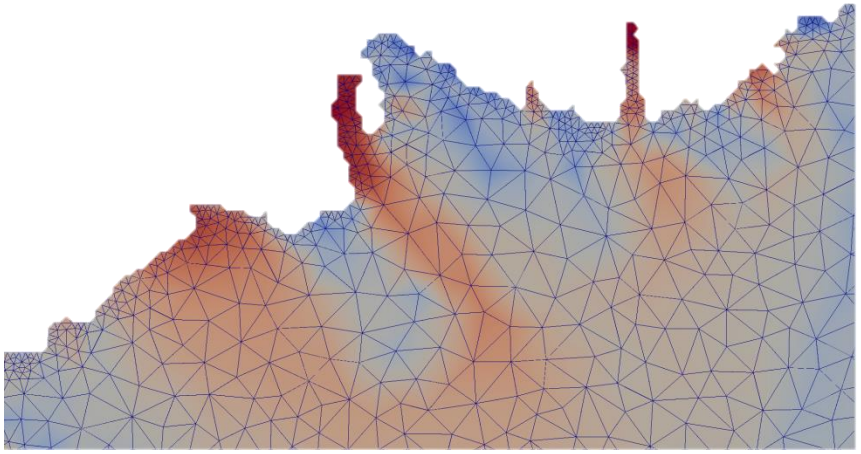
Hands-on Exercise Outline

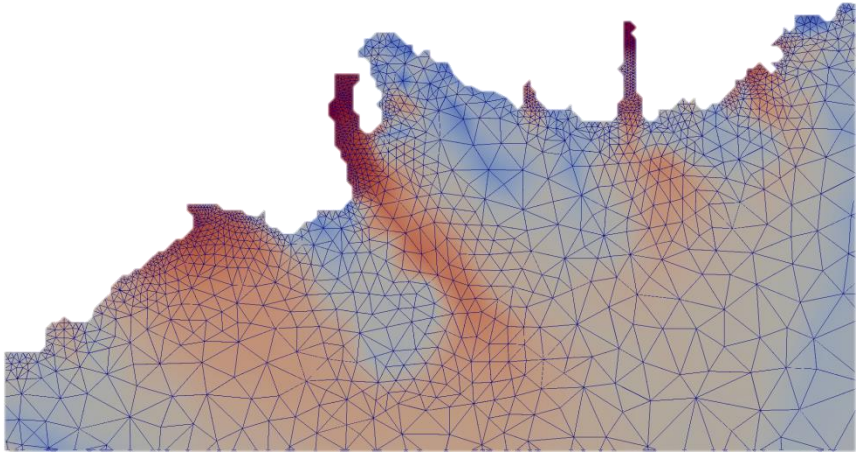
- Partition via Zoltan and ParMA
  - Multi-level graph and recursive inertial bisection
  - ParMA Vtx>Elm
- Visualization with ParaView



Hands-on Exercise Outline

- PAALS
  - In-memory parallel adaptive loop
    to analyze ice sheet flow
  - Running on 32 cores
  - Combining
    - Parallel mesh adaptation
      - Linear prism mesh elements
    - SPR-based error estimation
    - Local solution transfer of state variables
    - Predictive load balancing





- **Hands-on exercise**
  - https://github.com/gahansen/Albany/wiki/PAALS-Tutorial-2016
- **Capabilities:**
  - *Agile Component*-based, massively parallel solution adaptive multiphysics analysis
  - Fully-coupled, in-memory adaptation and solution transfer
  - Parallel mesh infrastructure and services
  - Dynamic load balancing
  - Generalized error estimation drives adaptation

PUMI Parallel Unstructured Mesh Infrastructure

Albany

ALEGRA SHOCK & MULTIPHYSICS

- **Download:**
  - Albany (http://gahansen.github.io/Albany)
  - SCOREC Adaptive Components (https://github.com/SCOREC)
- **Further information:** Mark Shephard [shephard@rpi.edu]
  Glen Hansen [gahanse@sandia.gov]