



FASTMath Unstructured Mesh Technologies

**Presenters: Mark S. Shephard, Vijay S. Mahadevan,
Glen Hansen and Cameron W. Smith**

FASTMath SciDAC Institute



Argonne National Laboratory

- Vijay Mahadevan
- Jim Jiao (Stony Brook)
- Paul Wilson (U. Wisconsin)

Lawrence Livermore National Lab.

- Barna Bihari
- Lori Diachin

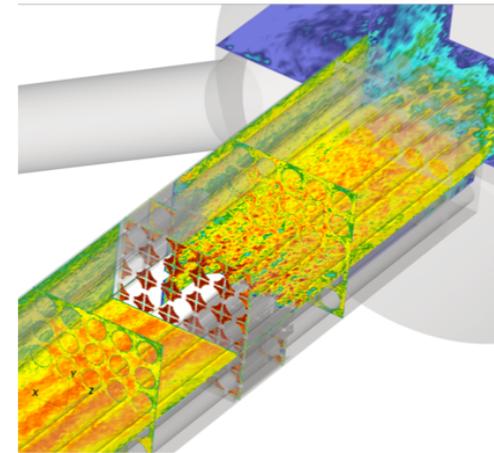
Sandia National Laboratories

- Andrew Bradley
- Karen Devine
- Glen Hansen
- 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)

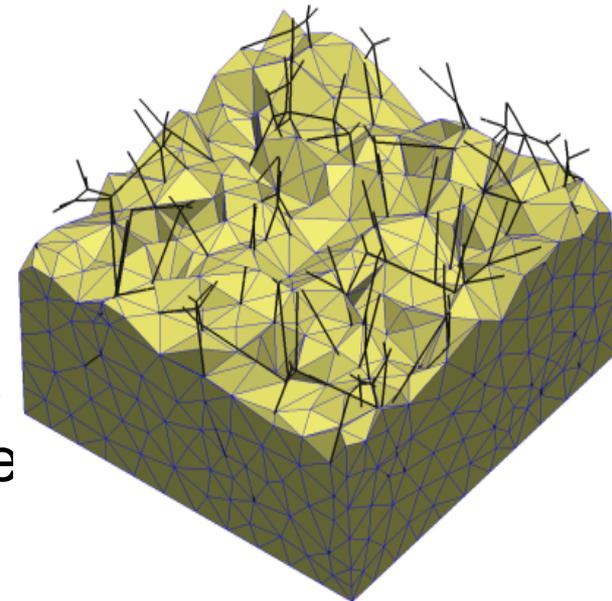
- Unstructured mesh methods and the need for unstructured mesh components for use by analysis code developers
- Core unstructured mesh components:
 - Parallel Mesh infrastructures
 - Mesh Generation, Adaptation, Optimization
 - Fields
 - Solution transfer
- Dynamic load balancing
- Unstructured mesh/solver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



Unstructured mesh – 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

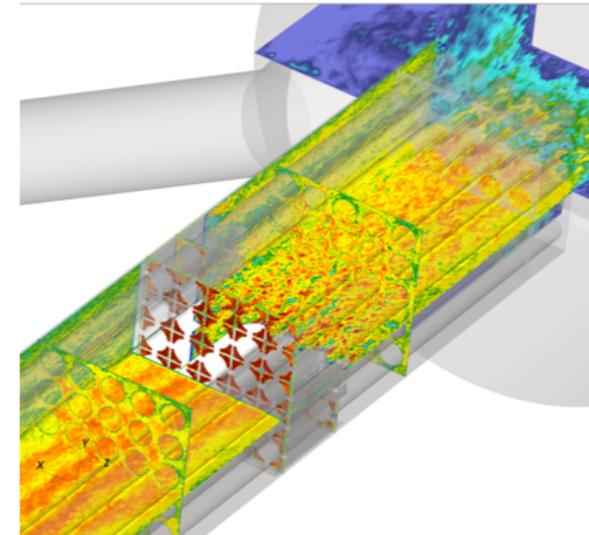
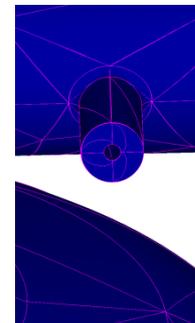
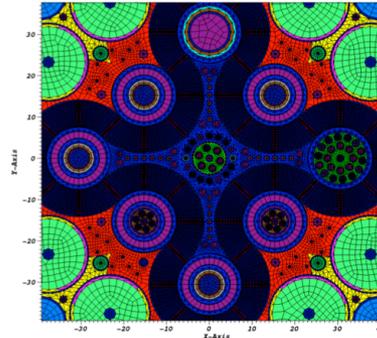


Disadvantages of unstructured meshes

- 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

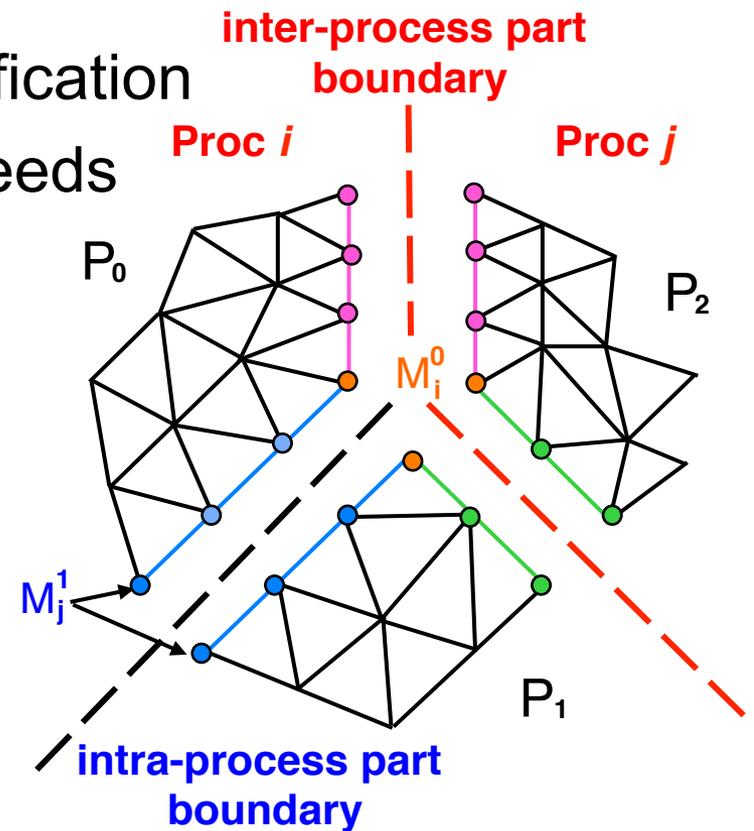
- Goal of FASTMath unstructured mesh developments include:
- Provide component-based tools that take full advantage of unstructured mesh methods and are easily used by analysis code developers and users
 - Develop those components to operate through multi-level APIs that increase interoperability and 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

- 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)
- IMEX FE Shock Multiphysics (Alegra/Alexa)



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

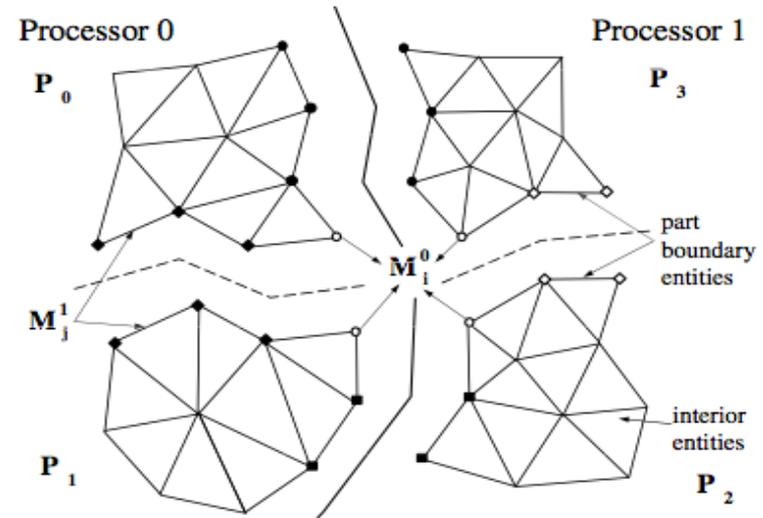


A 'part' is a set of mesh entities assigned to a process

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

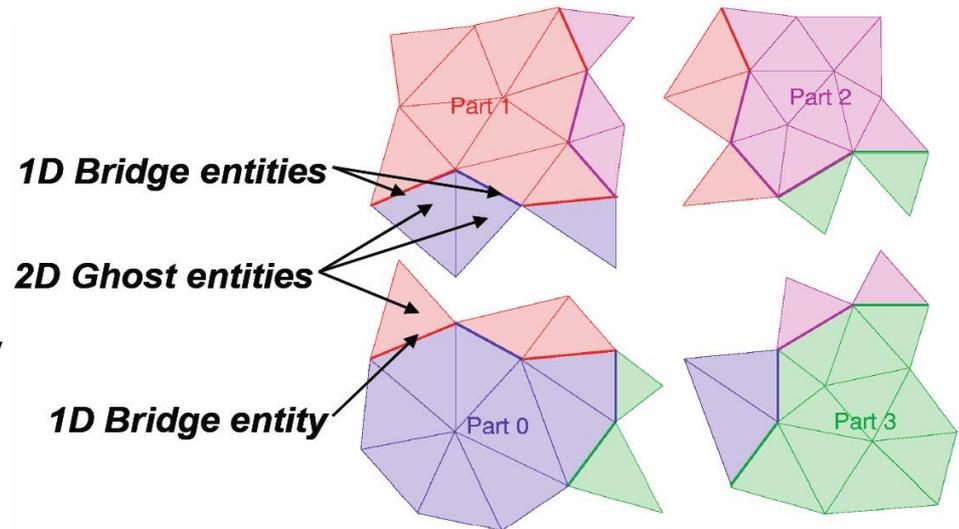
Mesh Migration

- 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



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

- 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

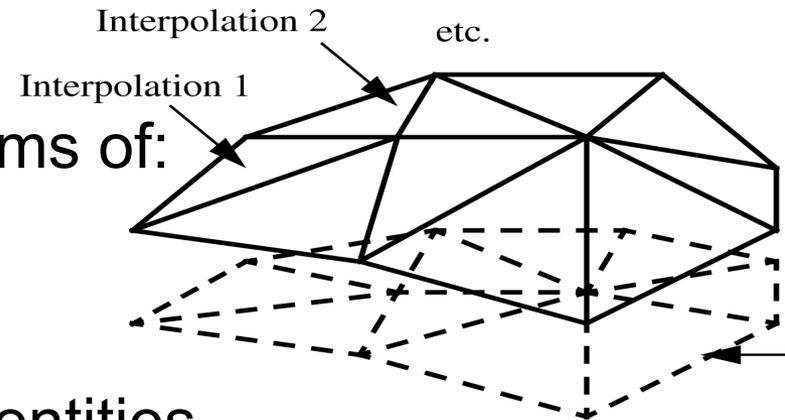
Mesh Shape Optimization

- 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

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.



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

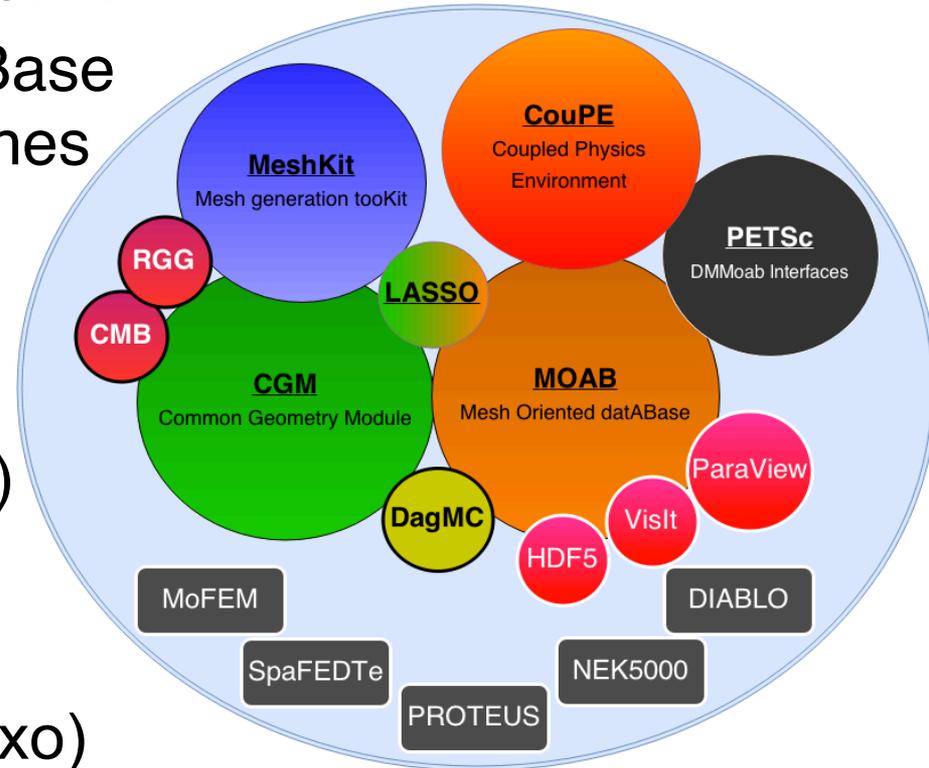
★ Solver interfaces

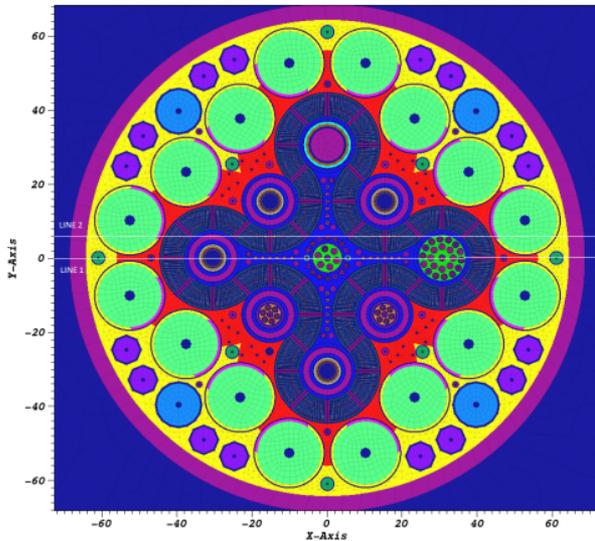
★ CouPE – Coupled multi-physics Environment

★ PETSc – MOAB (DMMoab) Discretization Manager

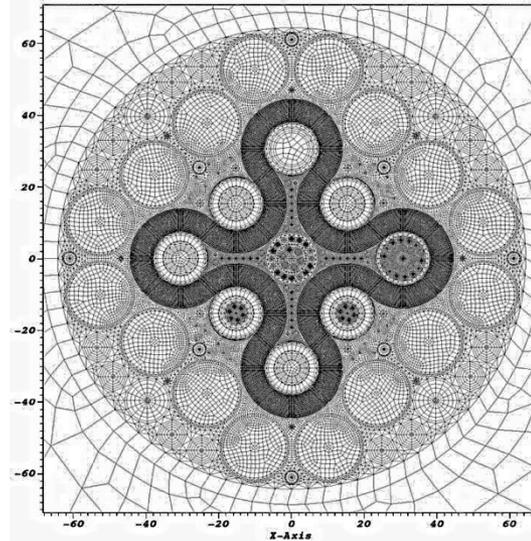
★ Scalable HDF5 serialization

★ In-situ visualization (h5m/vtk/exo)

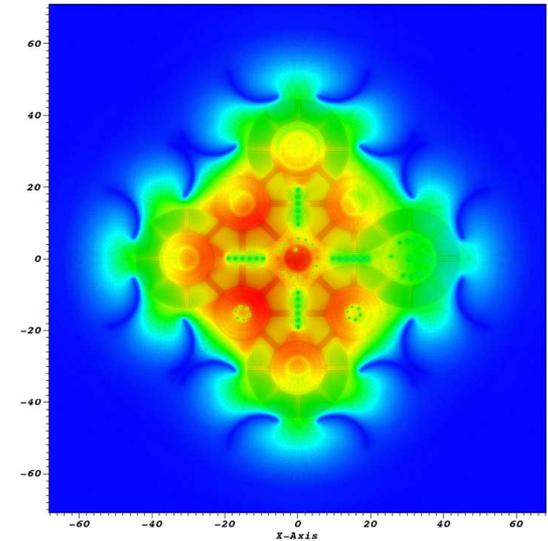




Geometry/BC Setup



Generate unstructured mesh
and link with solvers



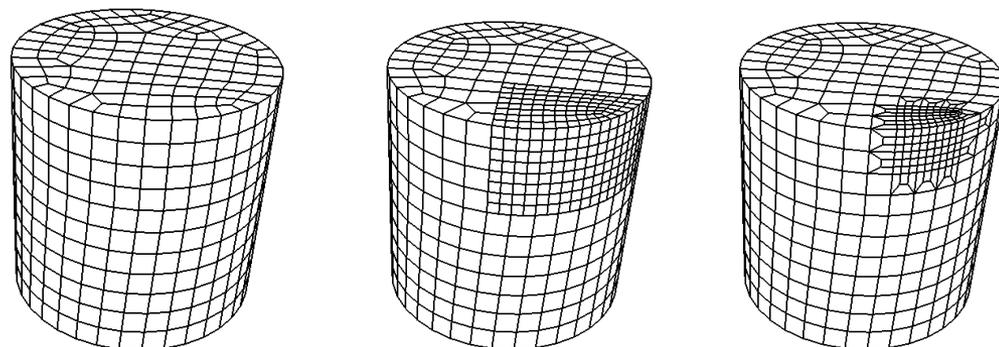
Check-point/Analyze/Visualize

- CGM: Geometry engine with interfaces to ACIS/OCC.
- MeshKit: Graph-based plugin design (**point** → **line** → **tri** → **tet**)
- Several efficient native algorithms and links to external mesh generation libraries (CUBIT, Netgen, Tetgen, CAMAL, GMsh)
- Accepts conventional **text (templates)** or **CAD** descriptions.

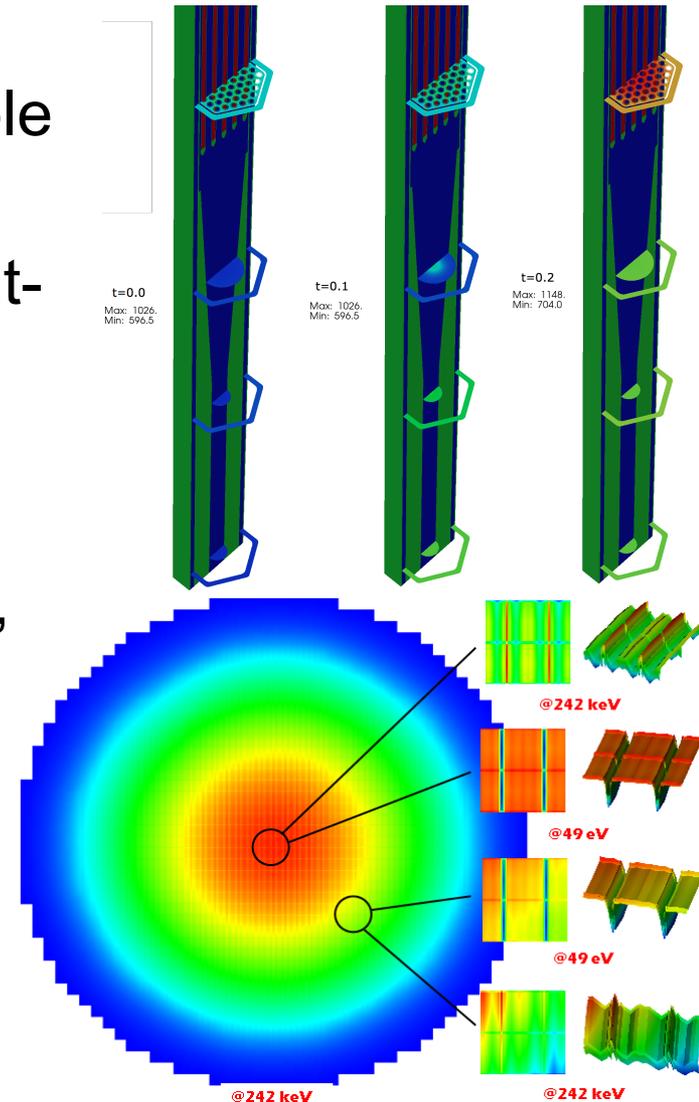
- Array-based unstructured mesh data-structure
 - Support stencil and block structured computations
 - Support thread safety for portable performance
- Dynamic load balancing: Zoltan, PARMetis
- Discretization kernels: cG, dG, Spectral, GFD, <user>
- Uniform mesh refinement hierarchy generation
 - Recover high order projection through 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.

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

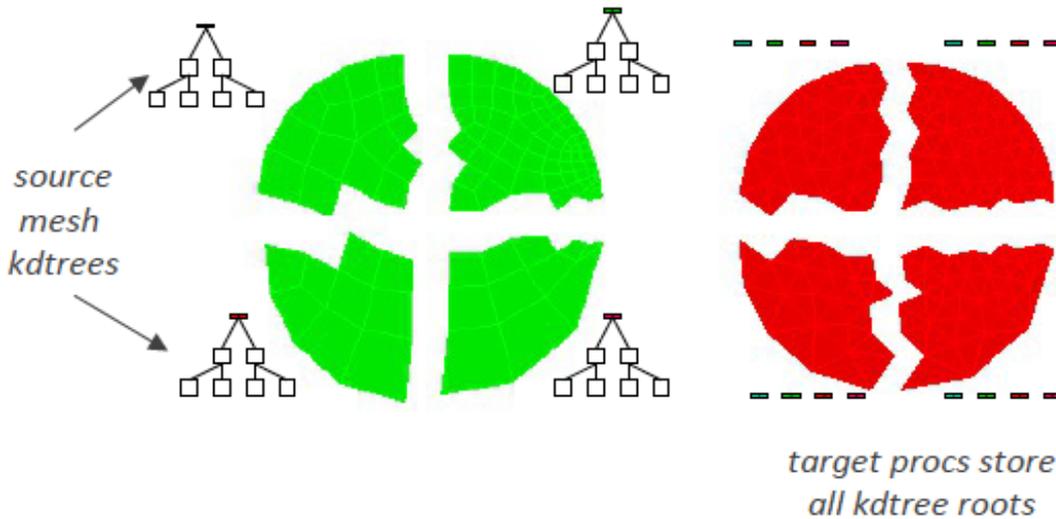
- a) Original
- b) Non-conformal
- c) Conformal



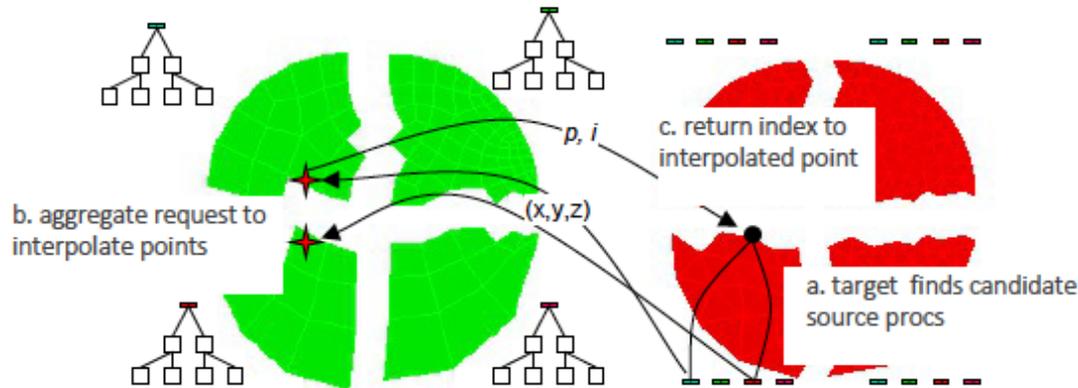
- Goals: Simplify geometry search and unify discretization kernels with a flexible interface for **coupled problems**.
- Geometry search: support parallel point-in-element query for various element topologies (edge, tri/quad/polygon, tet/hex/prism/pyramid)
- Discretization: support transformations, higher-order 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!



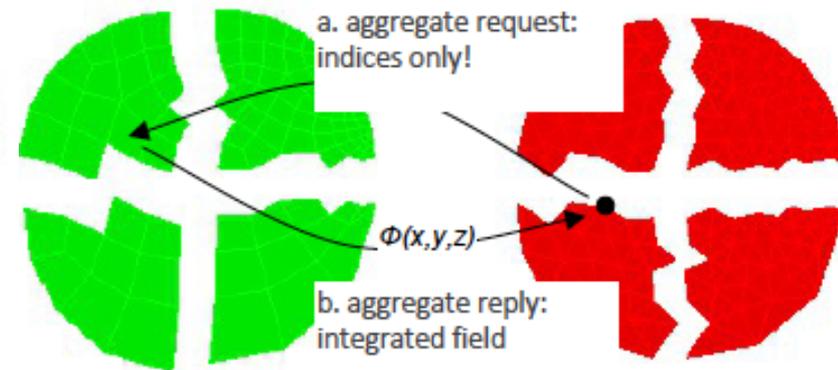
1. Initialization



2. Point Location



3. Interpolation

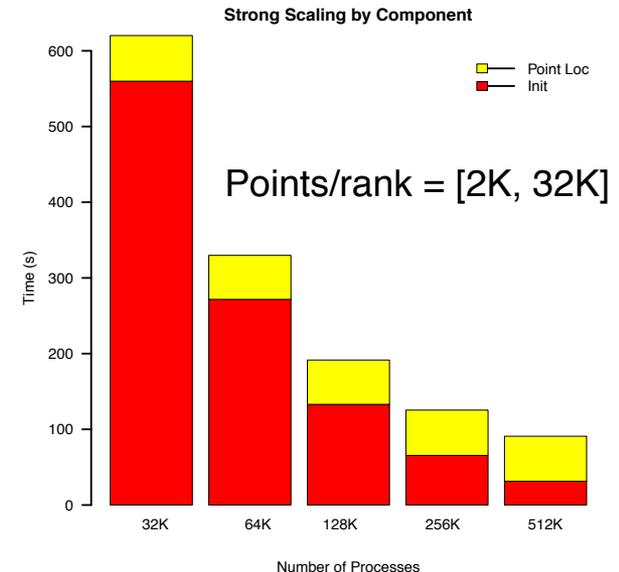
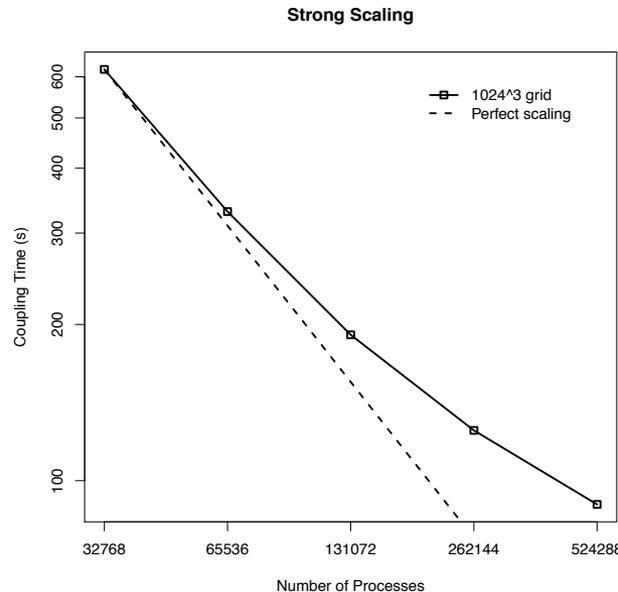
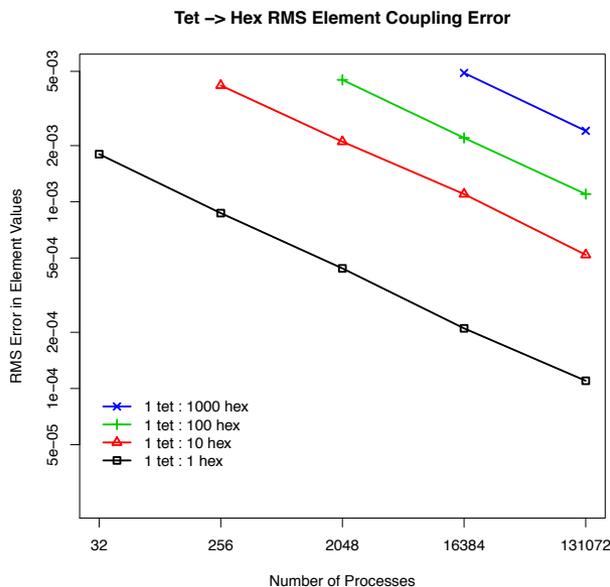


4. Normalization

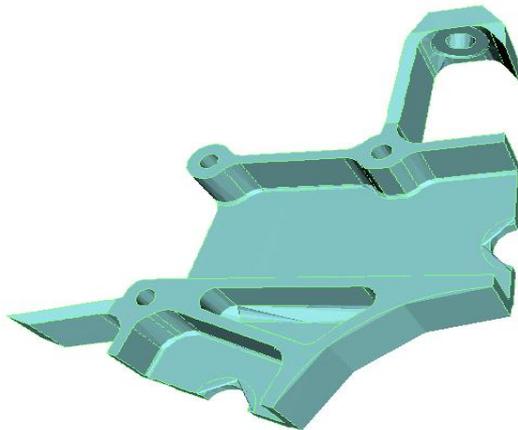
5. Conservation

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

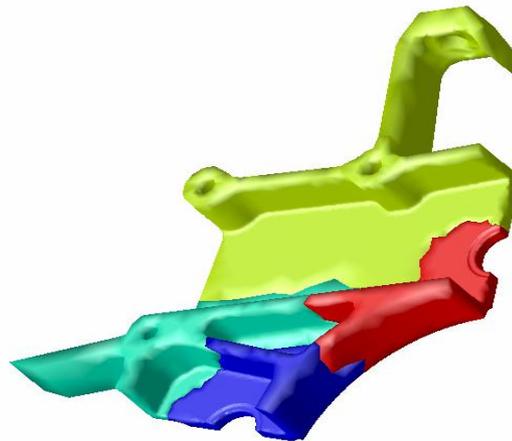
- Demonstrated 60% strong scalability of the solution transfer implementation in MOAB up to 512K cores on BG/Q.
- Bottleneck: Kd-tree scales as $O(n \log(n))$; Consider BVH/BIH trees to attain $O(\log(n))$ time complexity.
- Real problems: location vs interpolation, $O(1)$ vs $O(\Delta t)$
- Initialization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



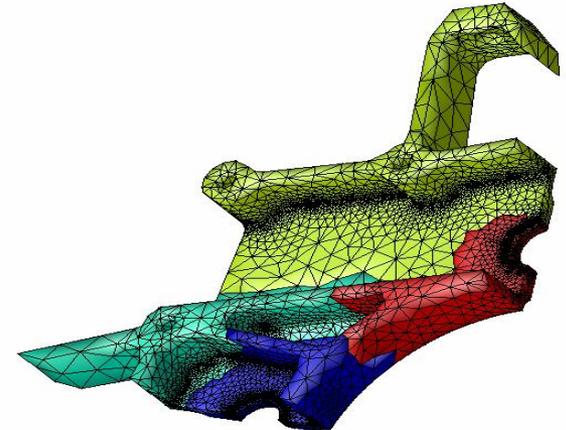
- Complete representation to provide any adjacency in $O(1)$ time
- Array-based storage for reduced memory size
- Parallel control through partition model that supports
 - All interprocess communications
 - Effective migration of mesh entities
 - Generalized read only copies



Geometric model

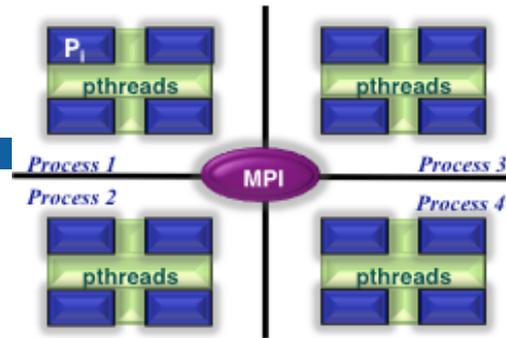


Partition model



Distributed mesh

- 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



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

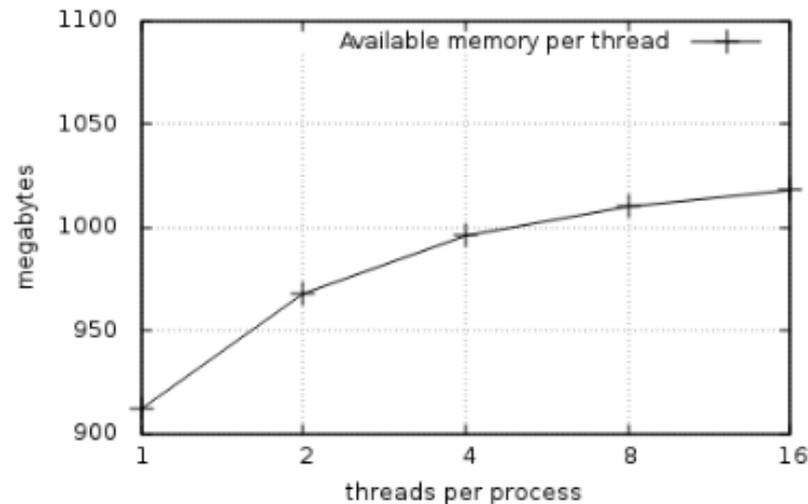
- Critical for many-core nodes where memory/core is limited

Use of Intel Phi accelerators

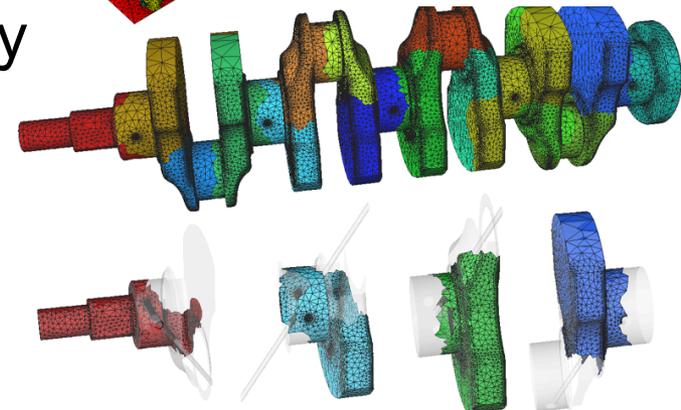
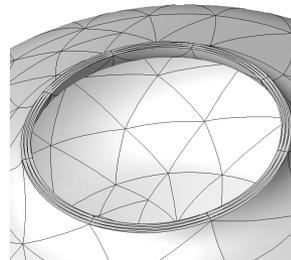
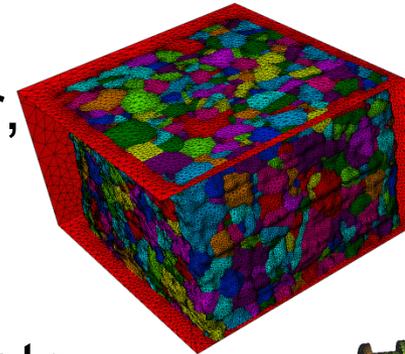
- On an equal number of Phi and BG/Q nodes

- 1024 → 2048 partitioning is 40% faster on Stampede
- 2048 → 4096 partitioning 8% slower on Stampede

Figure 6: Available memory with threads



- 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

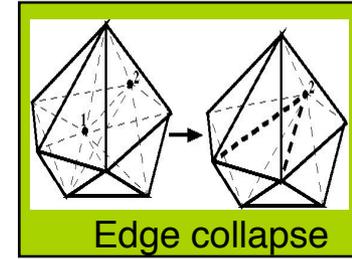
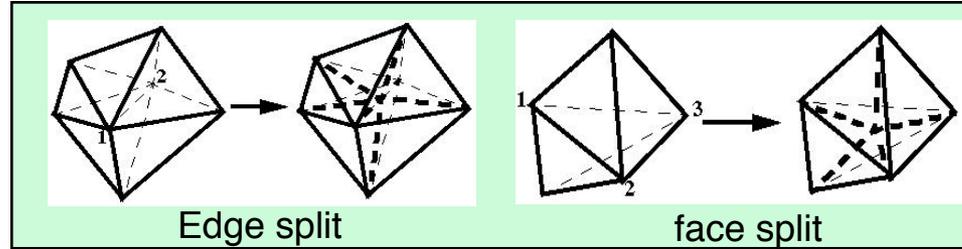


- Goal is the flexibility of remeshing with added advantages
- Strategy
 - Employ a “complete set” of mesh modification operations to alter the mesh into one that matches the given mesh size field
 - Driven by an anisotropic mesh size field that can be set by any combination of criteria
- Advantages
 - 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)

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

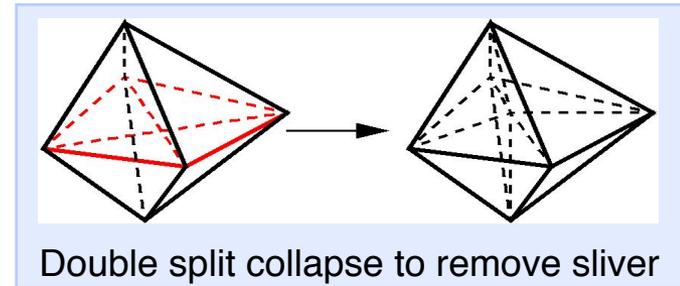
- Base operators

- Swap, collapse, split, move

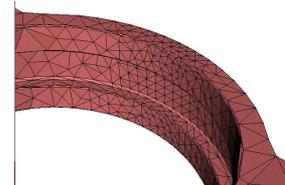
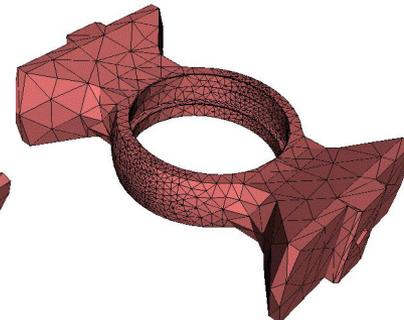
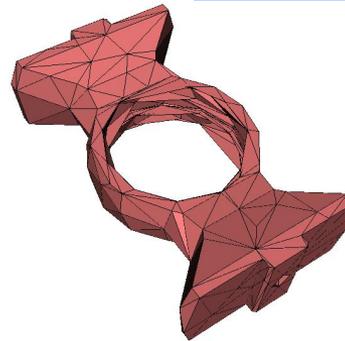


- Compound operators chain single step operators

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



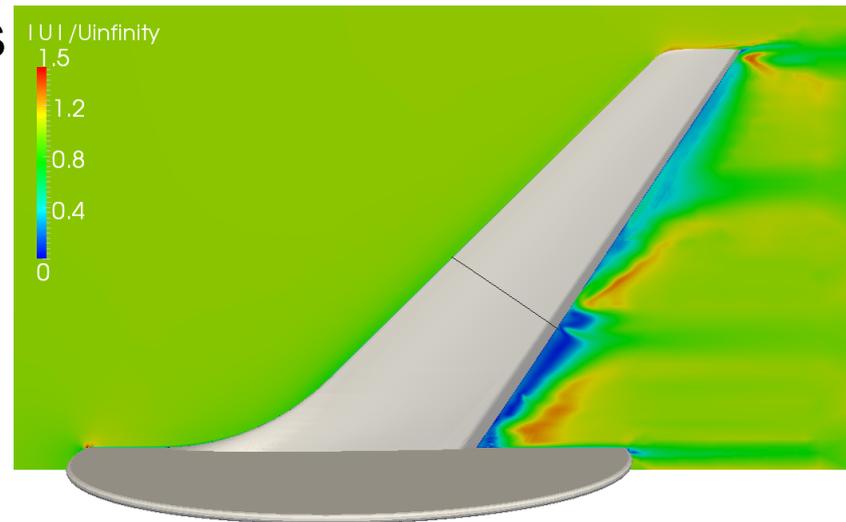
- Mesh adapts to true geometry



- Fully parallel

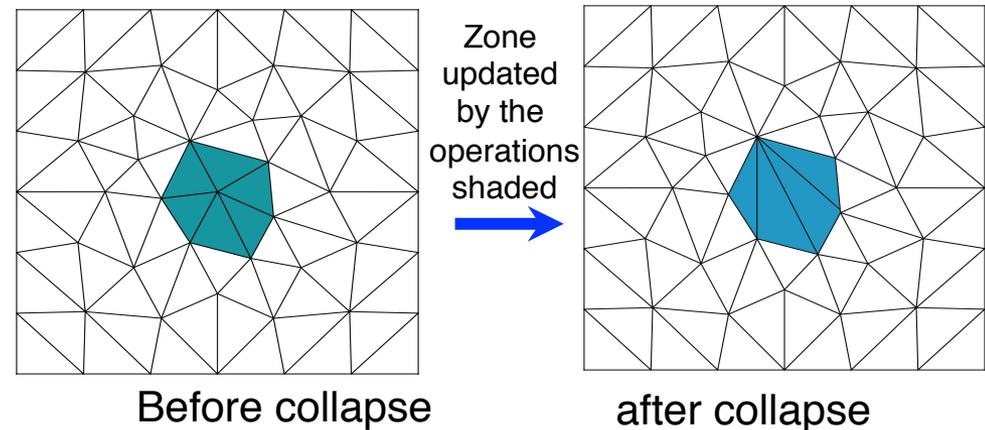
- Curved element geom.

- Attached Parallel Fields (APF) development underway
- 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

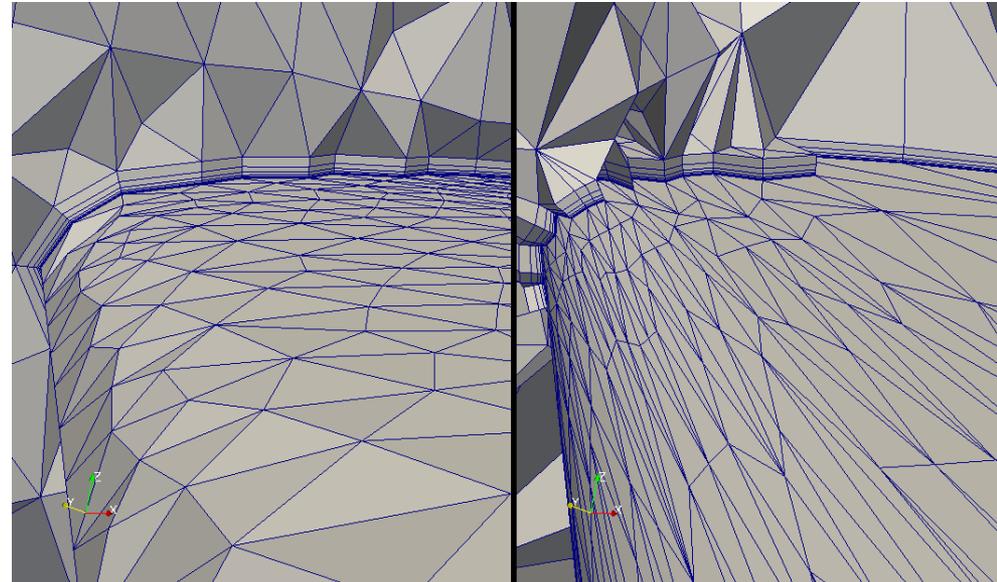
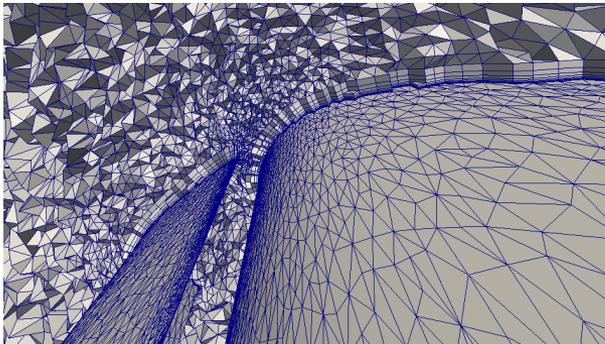
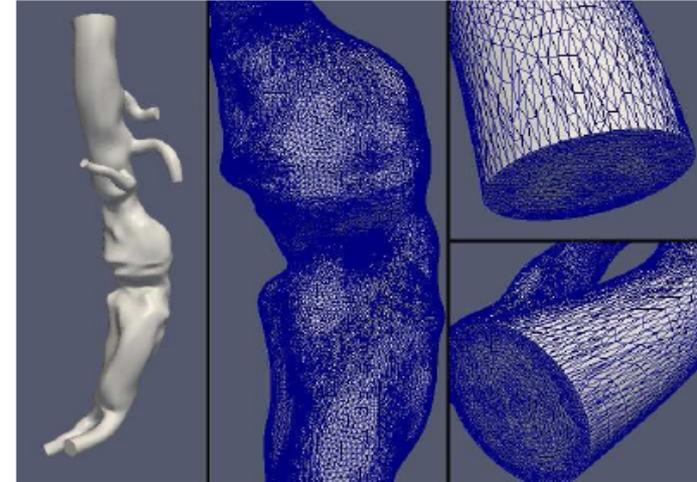


Local solution transfer during mesh adaptation

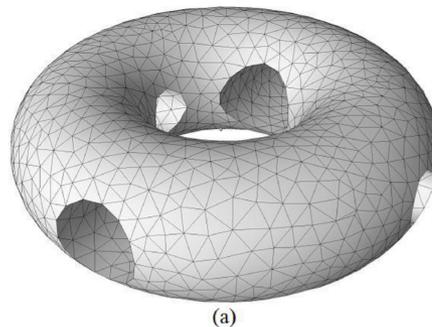
- Performed on cavity as local mesh modification performed
- Limited number of elements involved (no search over mesh)
- No accuracy loss with some operations (e.g., refinement)
- Others easier to control due to local nature (e.g., more accurate conservation correction)
- Applied to primary & secondary 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



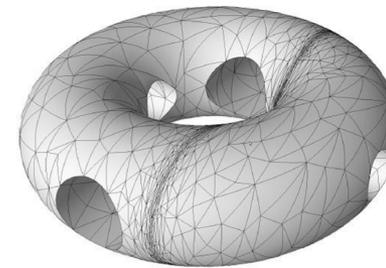
- Applied to very large scale models – 3.1M processes on $\frac{3}{4}$ million cores
- Local solution transfer supported through callback
- Effective storage of solution fields on meshes
- Supports adaptation with boundary layer meshes



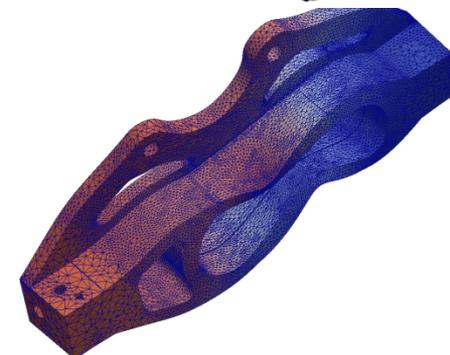
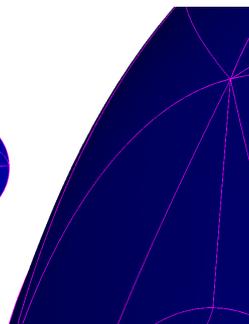
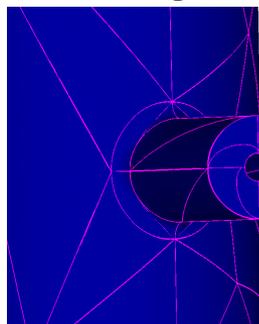
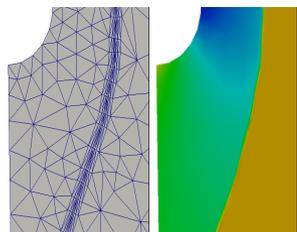
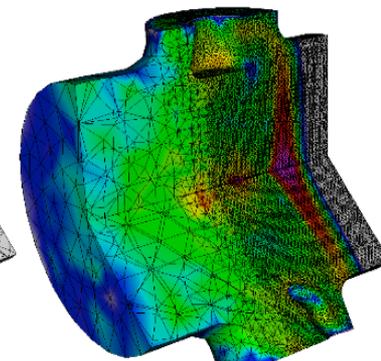
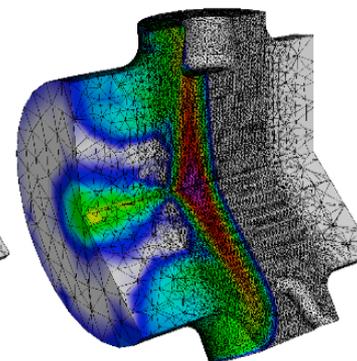
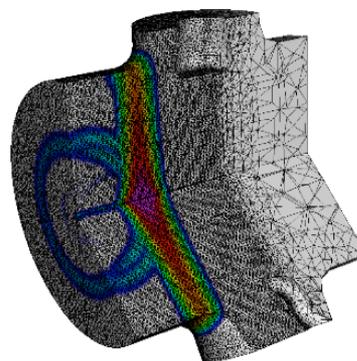
- Supports adaptation of curved elements
- Adaptation based on multiple criteria, examples
 - Level sets at interfaces
 - Tracking particles
 - Discretization errors
 - Controlling element shape in evolving geometry



(a)



(b)



- 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

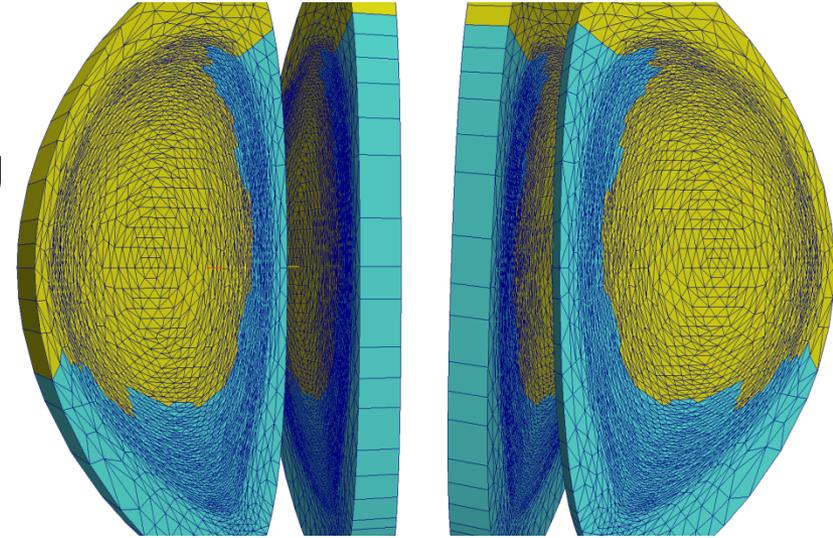
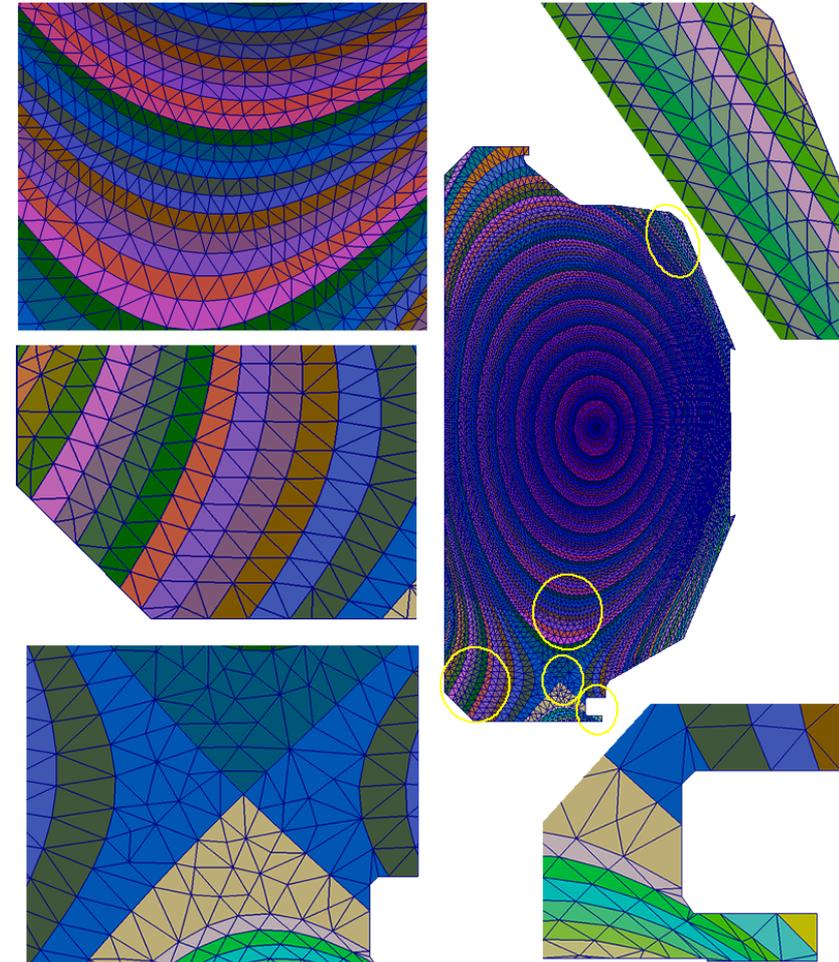


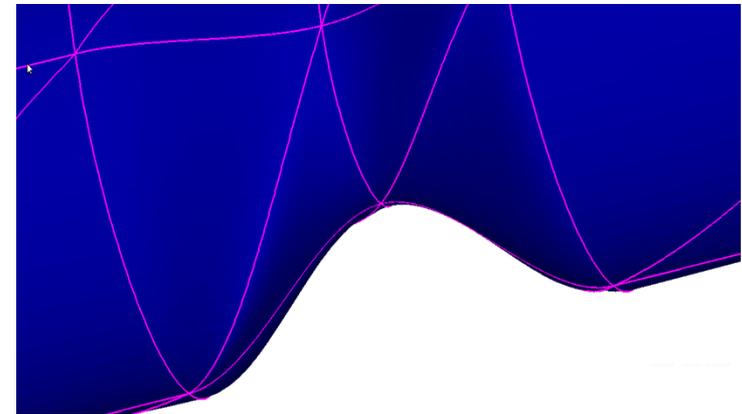
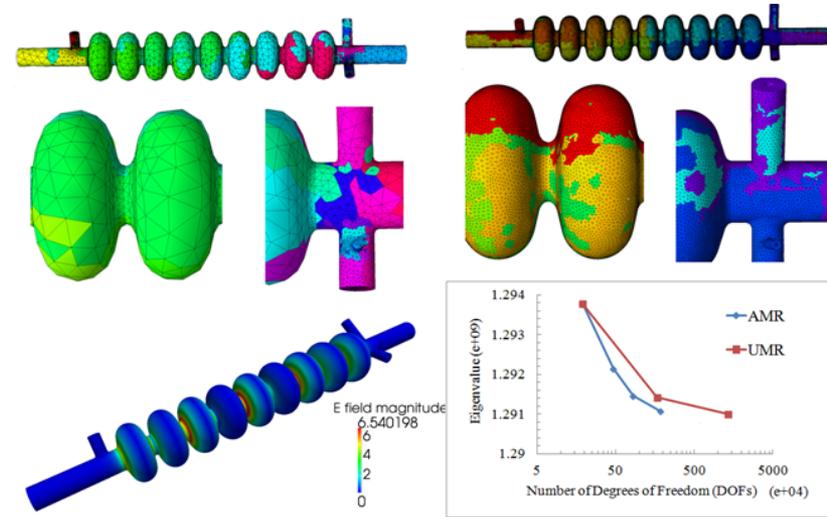
Fig: 3D mesh constructed from 64 2D planes on 12288 processes [1] (only the mesh between selected planes shown)

[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

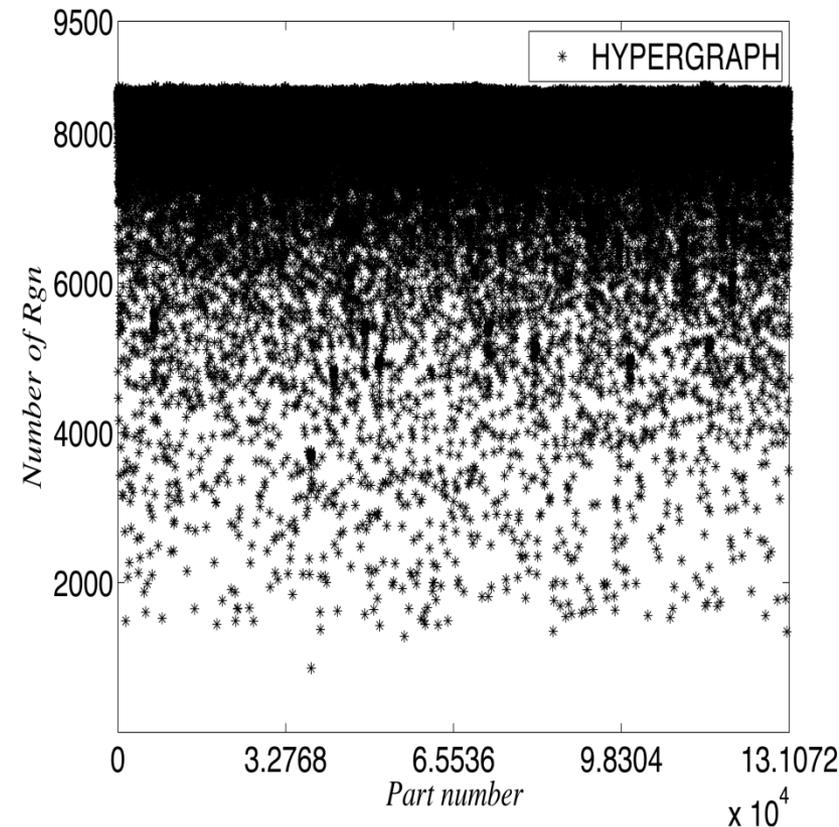
- EPSI PIC coupled to mesh simulation requires high quality meshes meeting a strict set of layout and other constraints
 - Previous method took >11 hours and mesh did not have desired quality
 - FASTMath meshing technologies put together to produce better quality meshes that meet constraints
 - Controlled 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

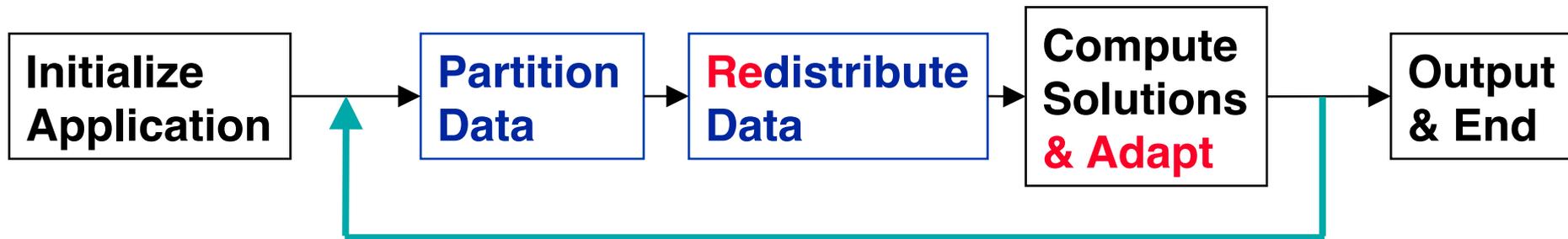


- Provide parallel mesh modification procedure capable of creating/adapting curved mesh geometry
- Parallel mesh adaptation procedure developed that supports quadratic curved meshes
- Ongoing efforts to support higher order G1 mesh geometry
- The procedure integrated with high-order electro-magnetic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



- 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 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.**

Static:

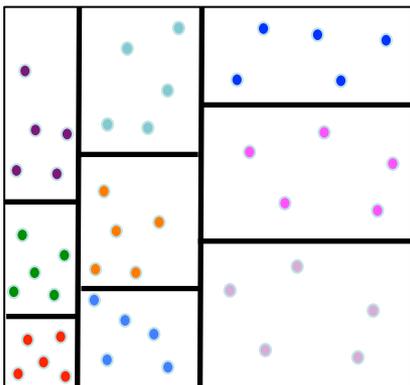
- Pre-processor to application.
- Can be implemented serially.
- May be slow, expensive.
- File-based interface acceptable.
- No consideration of existing decomposition required.

Dynamic:

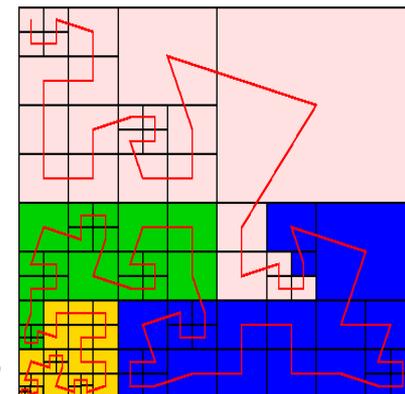
- Must run side-by-side with application.
- Must be implemented in parallel.
- Must be fast and scale.
- Library application interface required.
- Should be easy to use.
- Incremental algorithms preferred.
 - Small changes in input result in small changes in partitions.
 - Explicit or implicit incrementally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

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

Geometric

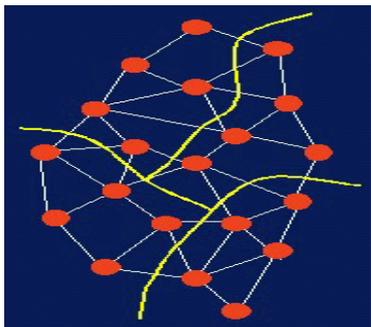


Recursive Coordinate Bisection
Recursive Inertial Bisection
Multi-Jagged Multi-section

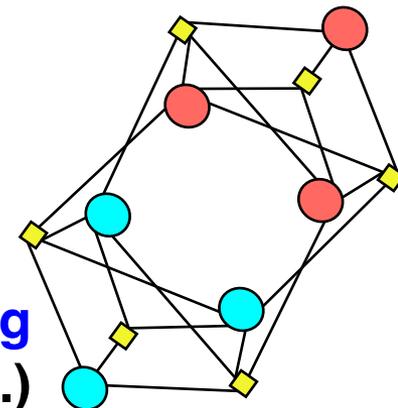


Space Filling Curves

Topology-based



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



PHG Hypergraph Partitioning
Interface to PaToH (Ohio St.)

Goal: Create parts containing physically close data

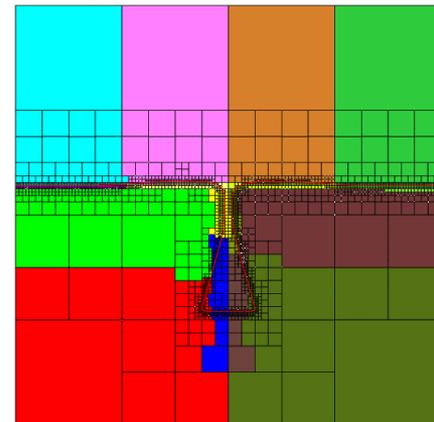
- RCB/RIB: Compute cutting planes that recursively bisect workloads
- MJ: Multi-section instead of bisection to reduce cost of partitioning
- SFC: Partition linear ordering given by space-filling curve

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



Goal: Balance work while minimizing data dependencies between parts

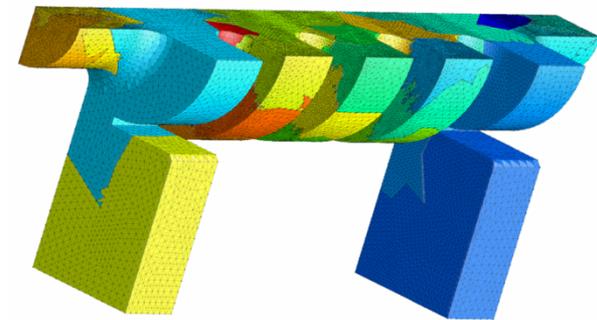
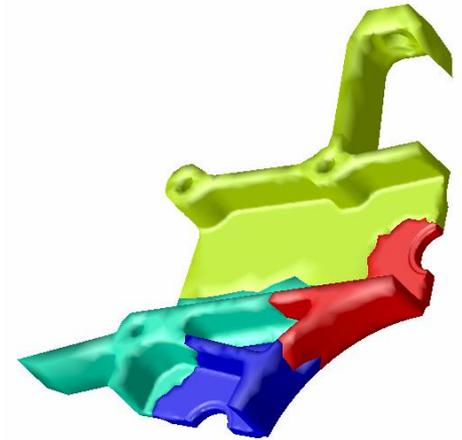
- Represent data with vertices of graph/hypergraph
- Represent dependencies with graph/hypergraph edges

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

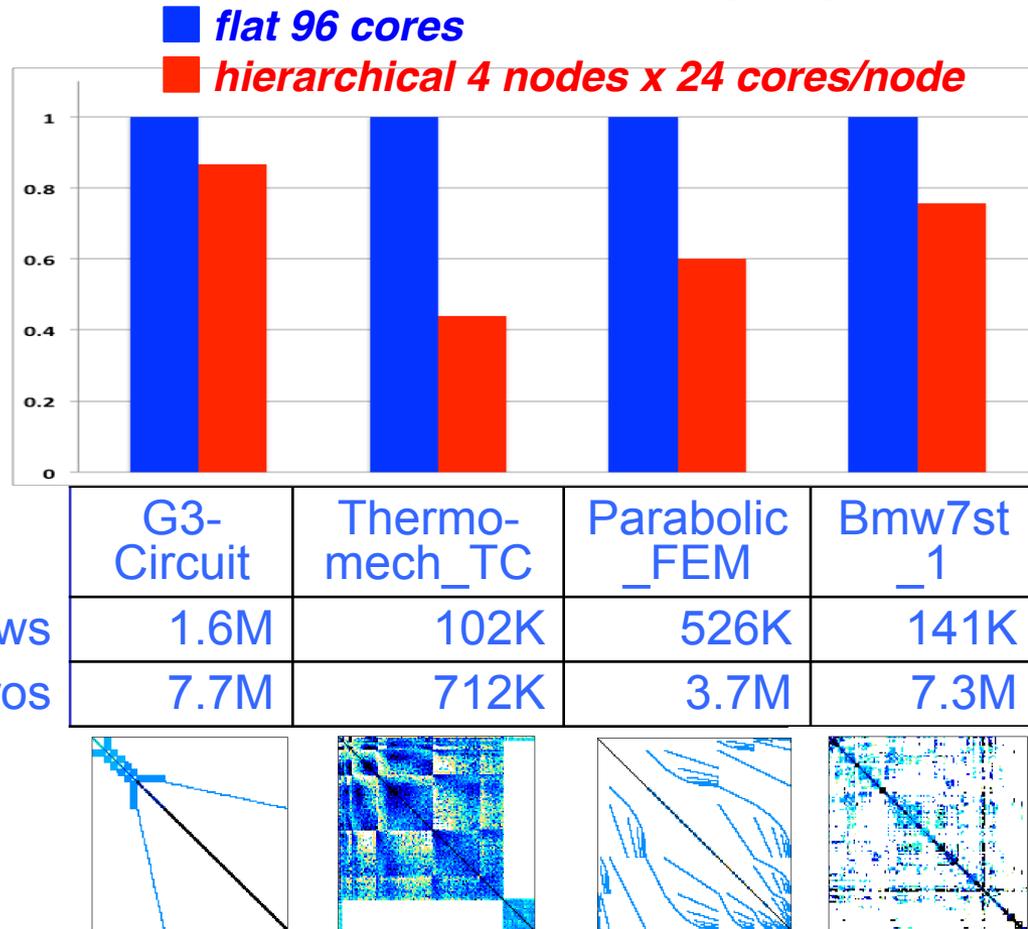
Disadvantages:

- More expensive than geometric approaches
- Require explicit dependence info



- 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

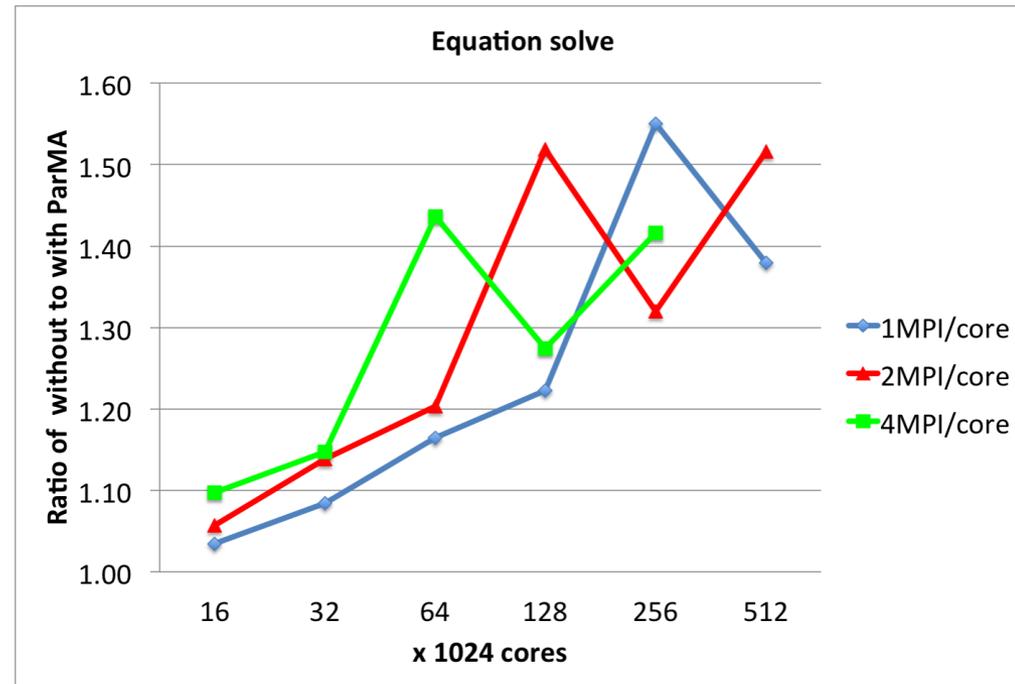


Mesh and partition model adjacencies directly used

- 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

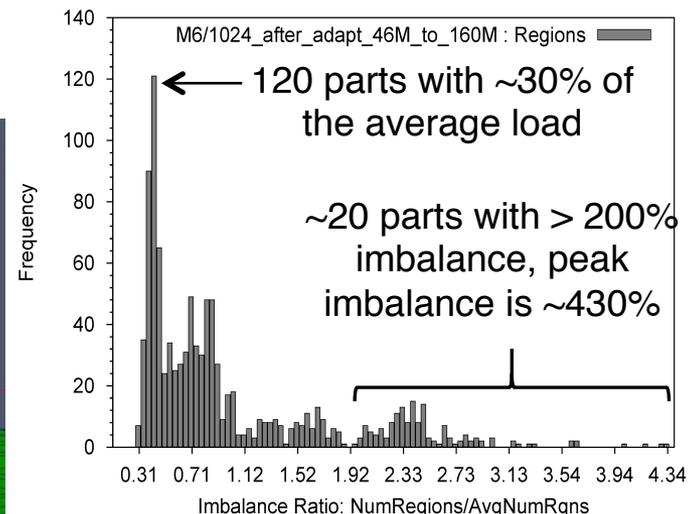
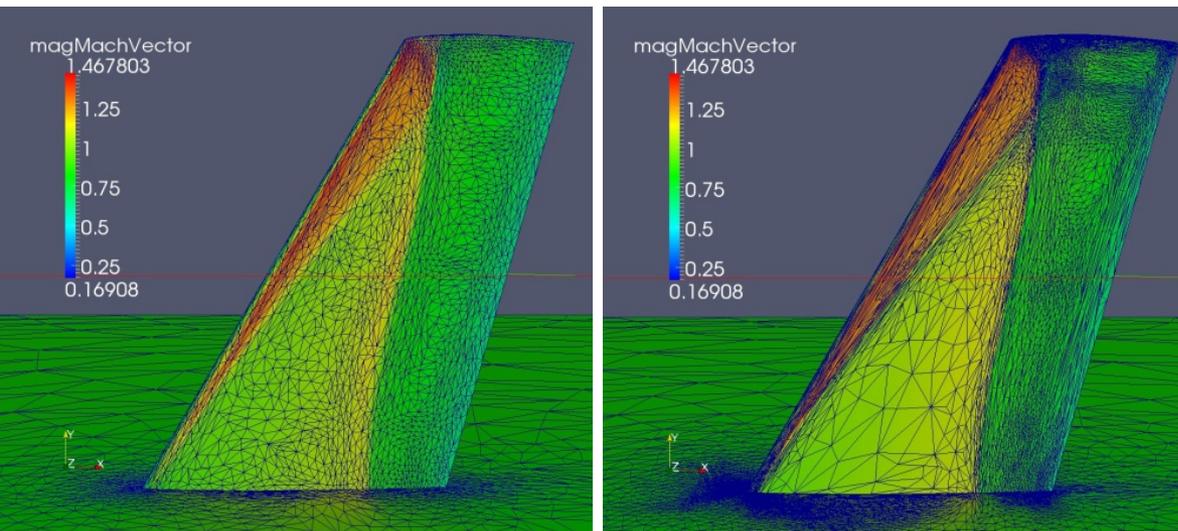
Example of C0, linear shape function finite elements

- 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.

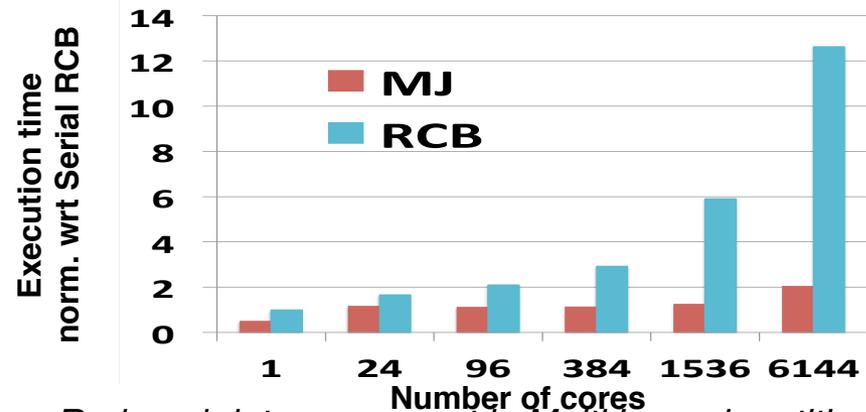
- Mesh modification before load balancing can lead to memory problems - common to see 400% increase on some parts
- Employ predictive load balancing to avoid the problem
 - Assign weights based on what will be refined/coarsened
 - Apply dynamic load balancing using those weights
 - Perform mesh modifications



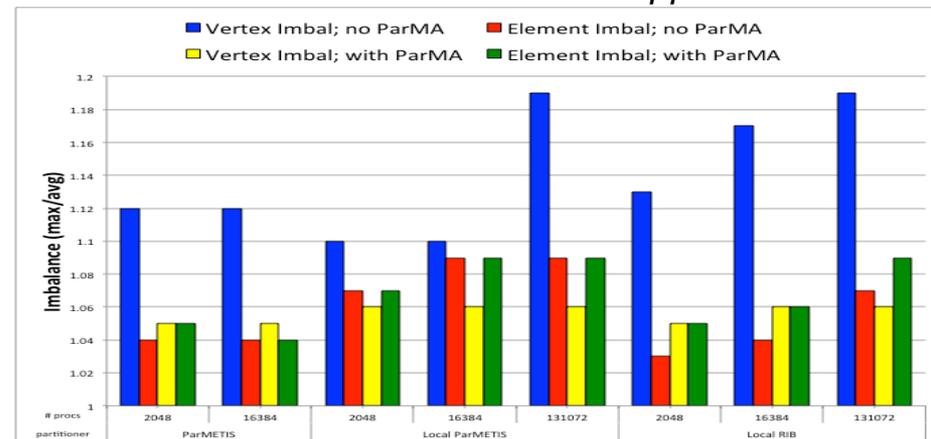
Histogram of element imbalance in 1024 part adapted mesh on Onera M6 wing if no balancing applied prior to adaptation.

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 $\frac{3}{4}$ 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.

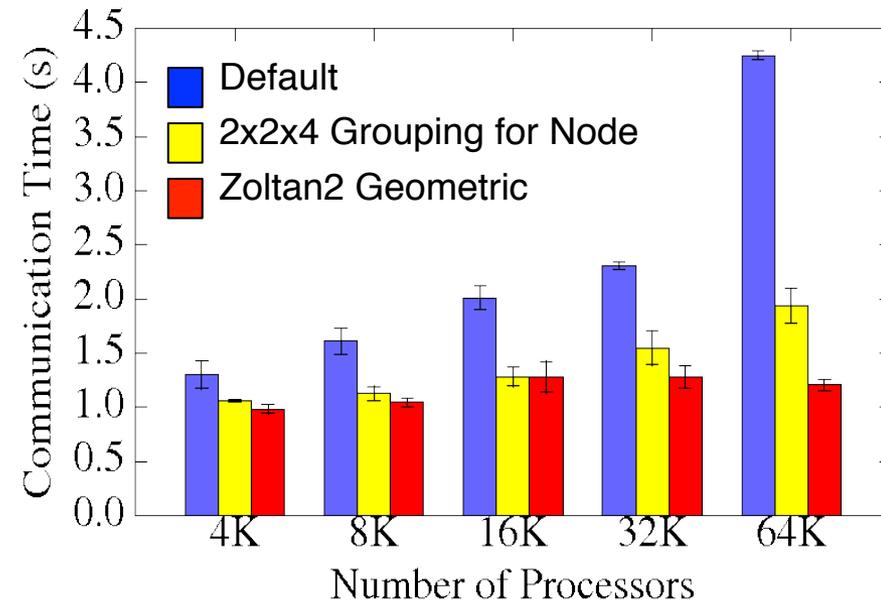
- Goal: Assign MPI tasks to cores so that application communication costs are low
- Especially important in non-contiguous node allocations (e.g., Hopper Cray XE6)

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



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

Need uniform interface to solve multi-component problems on unstructured meshes with FD/FEM/FVM on both structured and unstructured meshes.

- ⦿ Create 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.
- ⦿ Provide routines to **build simple meshes** in-memory or load an unstructured grid from file.
- ⦿ Analyze efficient unstructured **mesh traversal, FD/FEM-type operator assembly** for relevant multi-dimensional, multi-component problems.

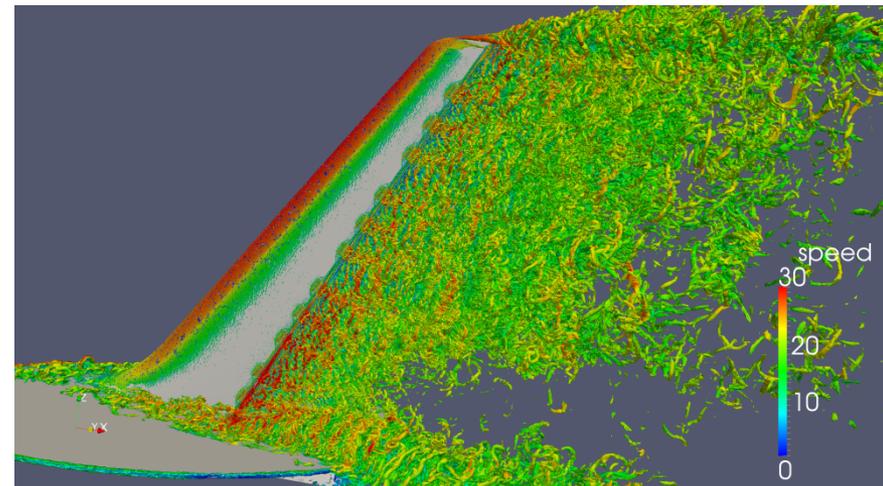
- ⊙ Design resembles structured (**DMDA**) and unstructured (**DMPlEx**) interfaces; software productivity.
- ⊙ Support both strided and interleaved access of field components; Opens up better preconditioning strategies.
- ⊙ Provide a uniform interface to solve nonlinear problems with FEM/FDM on both 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.

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

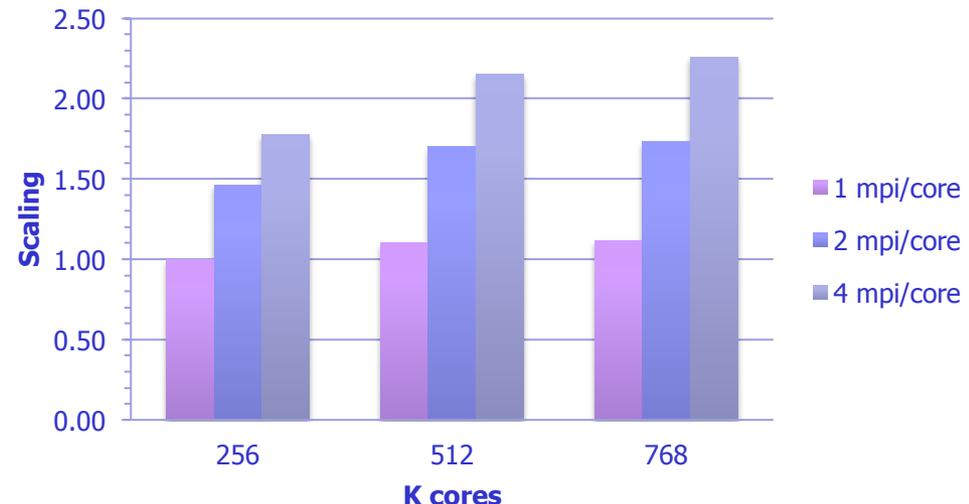
- ⊙ Reduce total memory usage by sharing vector spaces and allowing block filling of coupled component terms.
- ⊙ Fast stiff ODE-solvers for reaction-diffusion equations via IMEX methods (PETSc) that accommodate sparse coupling in high-order multi-component solvers.
- ⊙ Some relevant tutorial examples in PETSc:
 - ★ Multi-component time-dependent [Brusselator reaction-diffusion PDE FEM solver in 1-d.](#) (ts/examples/ex35.cxx)
 - ★ 2-D/3-D, [verifiable Diffusion-Reaction FEM steady state solver](#) with geometric multigrid. (ksp/examples/ex35.cxx and ksp/examples/ex36.cxx)
 - ★ 2-D Generalized Finite Difference poisson solver with GMG (ts/examples/GFD/ex2.cxx).
 - ★ 2-D Bratu (Solid Fuel Ignition) nonlinear PDE solver (snes/examples/ex36.cxx)

Implicit, Adaptive Grid CFD

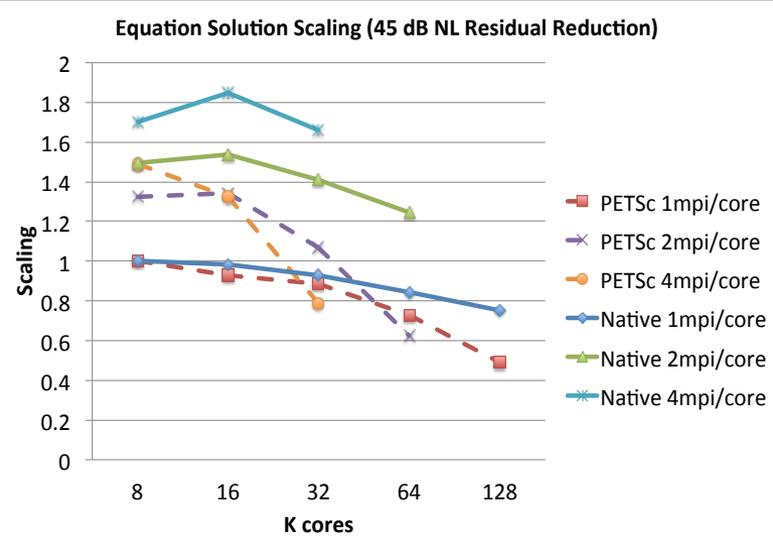
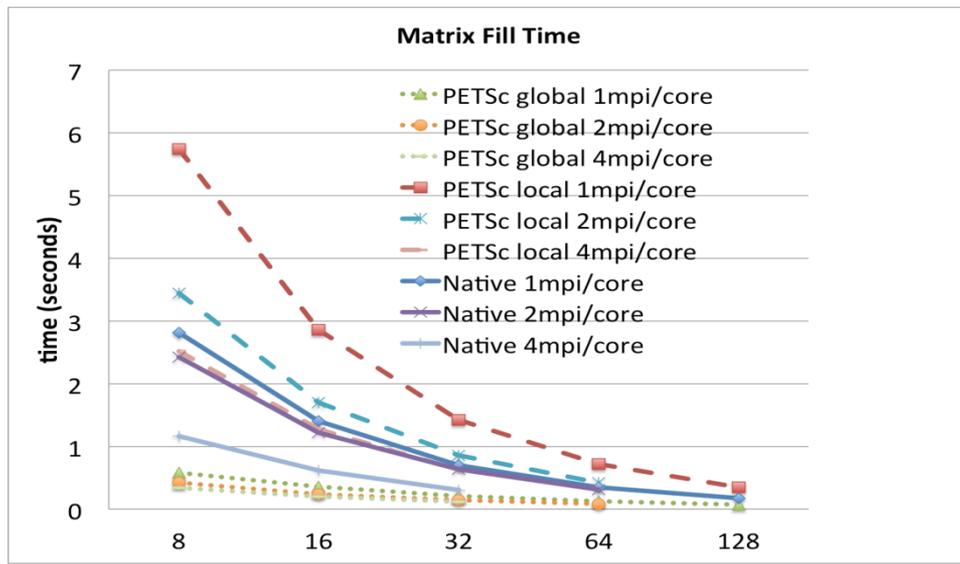
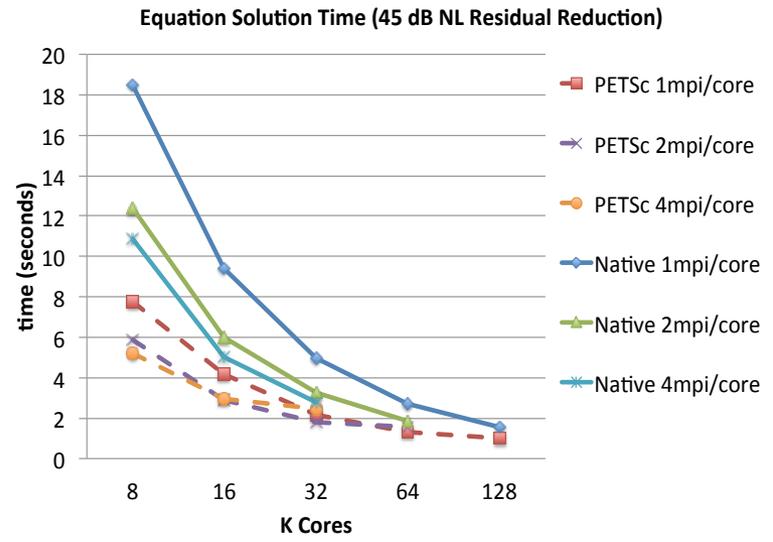
- Extreme Scale Applications:
 - Aerodynamics flow control
 - Multiphase flow
- Full Machine Strong scaling
 - Variable MPI processes/core
 - 92 Billion tetrahedra
 - 262144 to 3,145,728 parts
 - 1/core 100% scaling
 - 2/core 146-155% scaling
 - 4/core 178-226% scaling



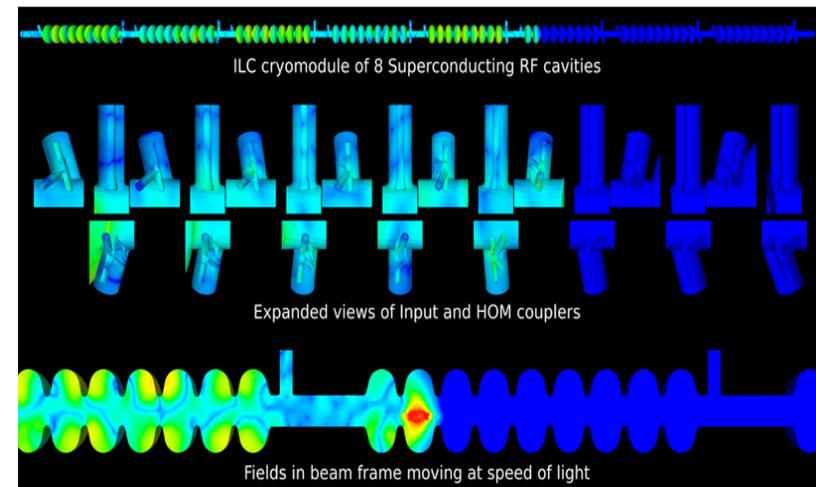
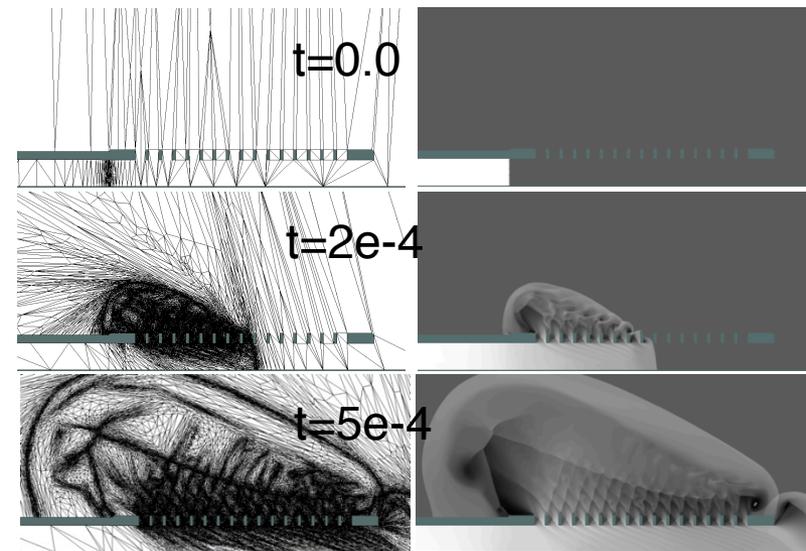
92 billion tetrahedra



- 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

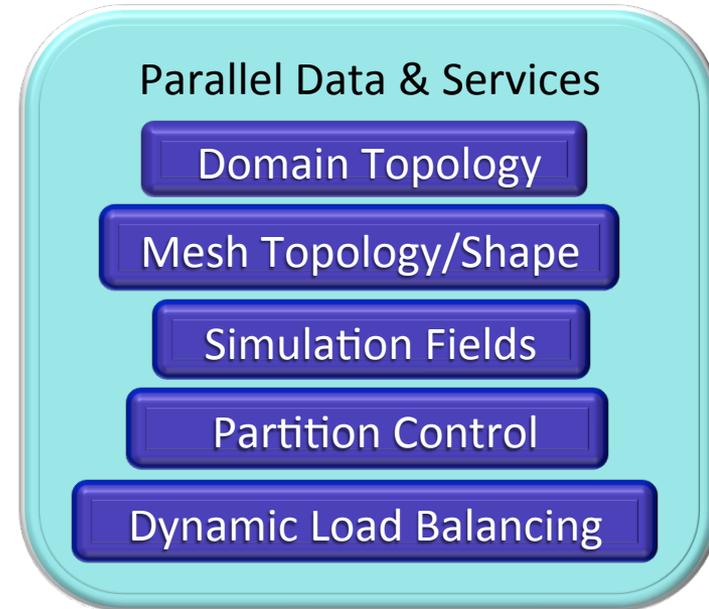


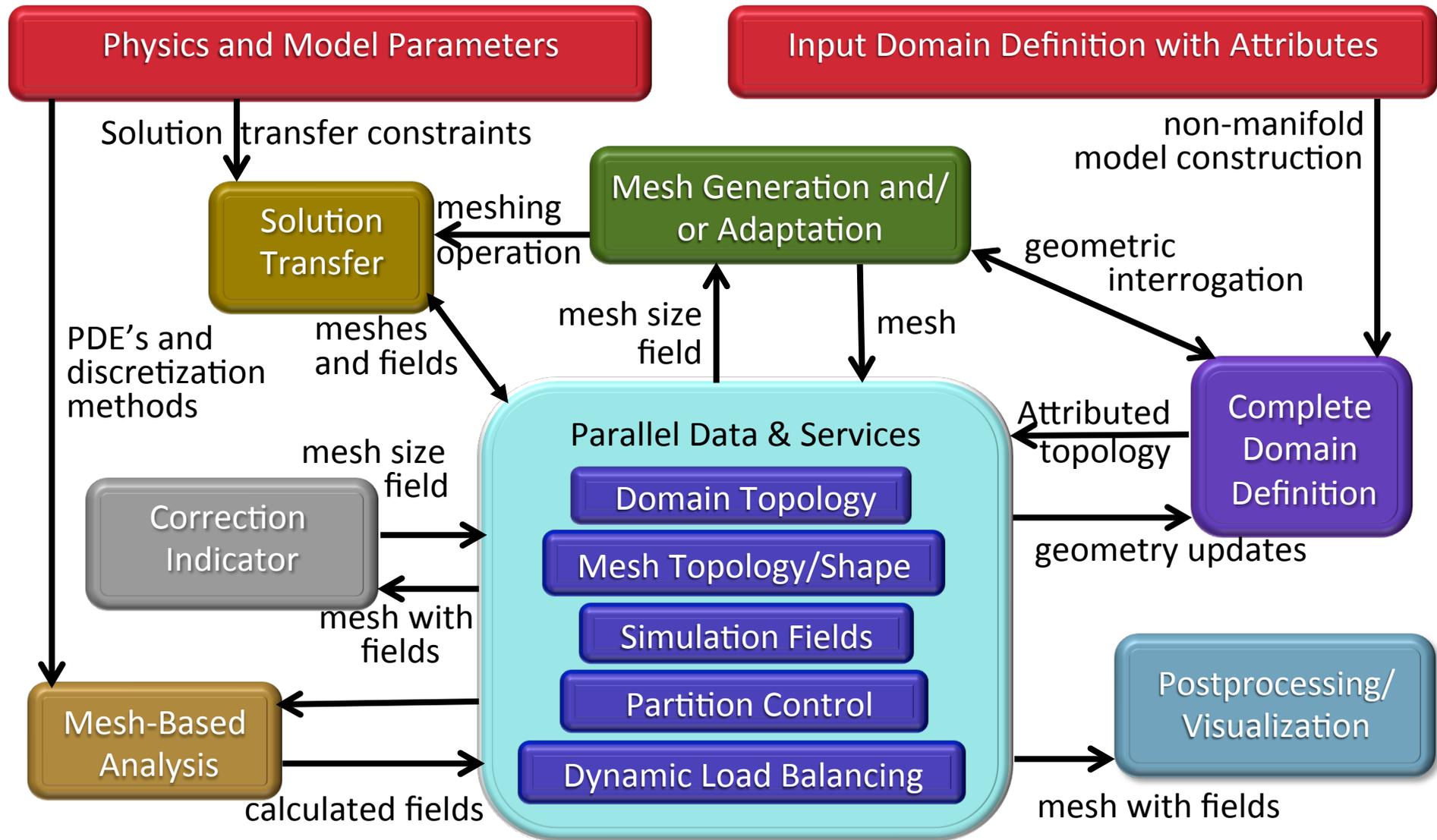
- Automation and adaptive methods critical to reliable simulations
- Users want flexibility to apply best in class analysis codes
 - Component-based approach to integrate automated adaptive methods with analysis codes
 - All components operate in parallel, including fast in-memory coupling
- Developing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



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



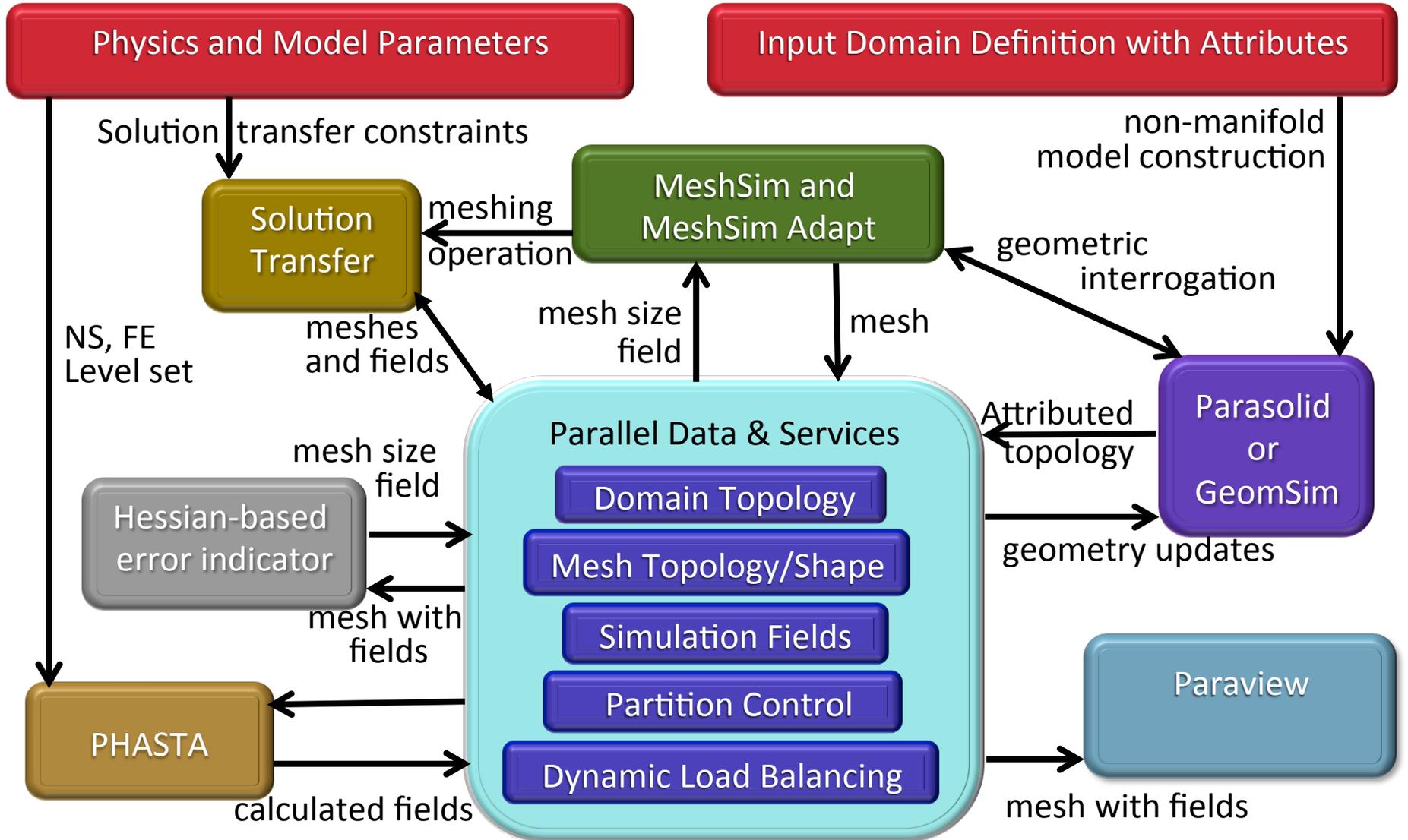


Approaches to avoid significant bottleneck of file based coupling

- Serialized data streams using existing file reading and writing protocol
 - Minimal code changes - ideal for components already coupled via files
- API based transfer procedures
 - Flexibility to extend functionality - ideal for components with APIs that support querying and modifying data structures

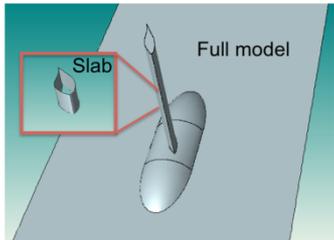
In-memory has far superior parallel performance

Number of processes	File-based elapsed time (sec)	In-memory elapsed time (sec)
512	49	2
2048	91	1



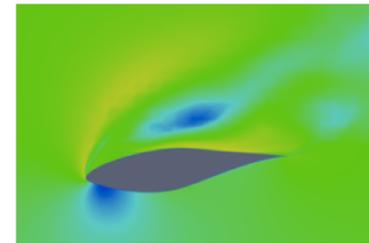
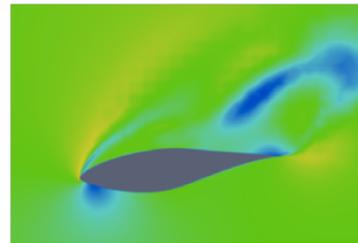
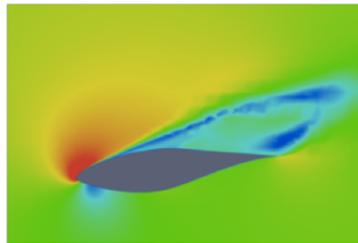
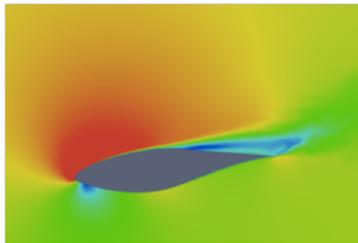
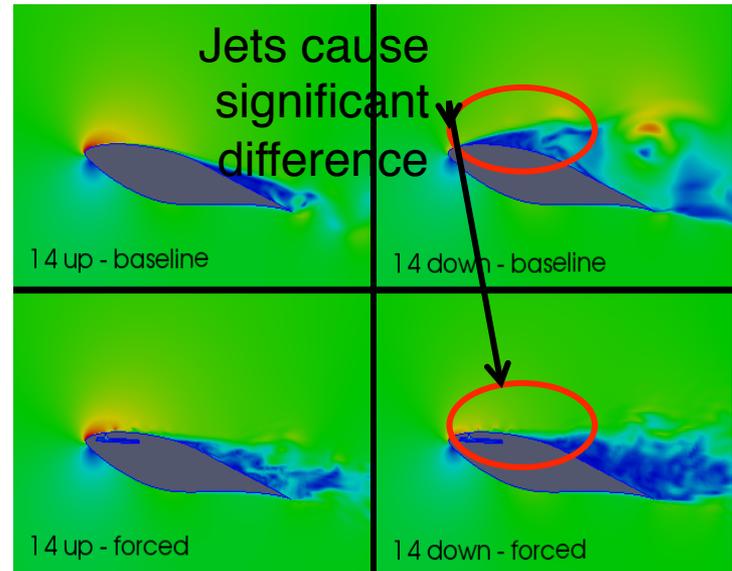
Dynamic pitch with angle of attack of $14^\circ \pm 5.5^\circ$

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

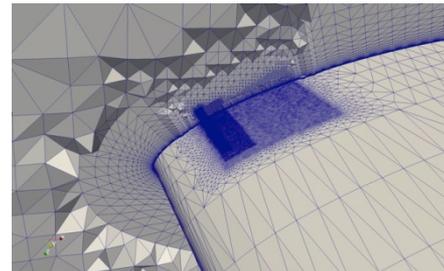
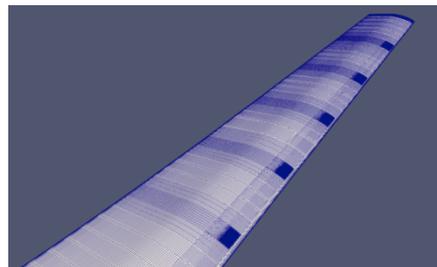


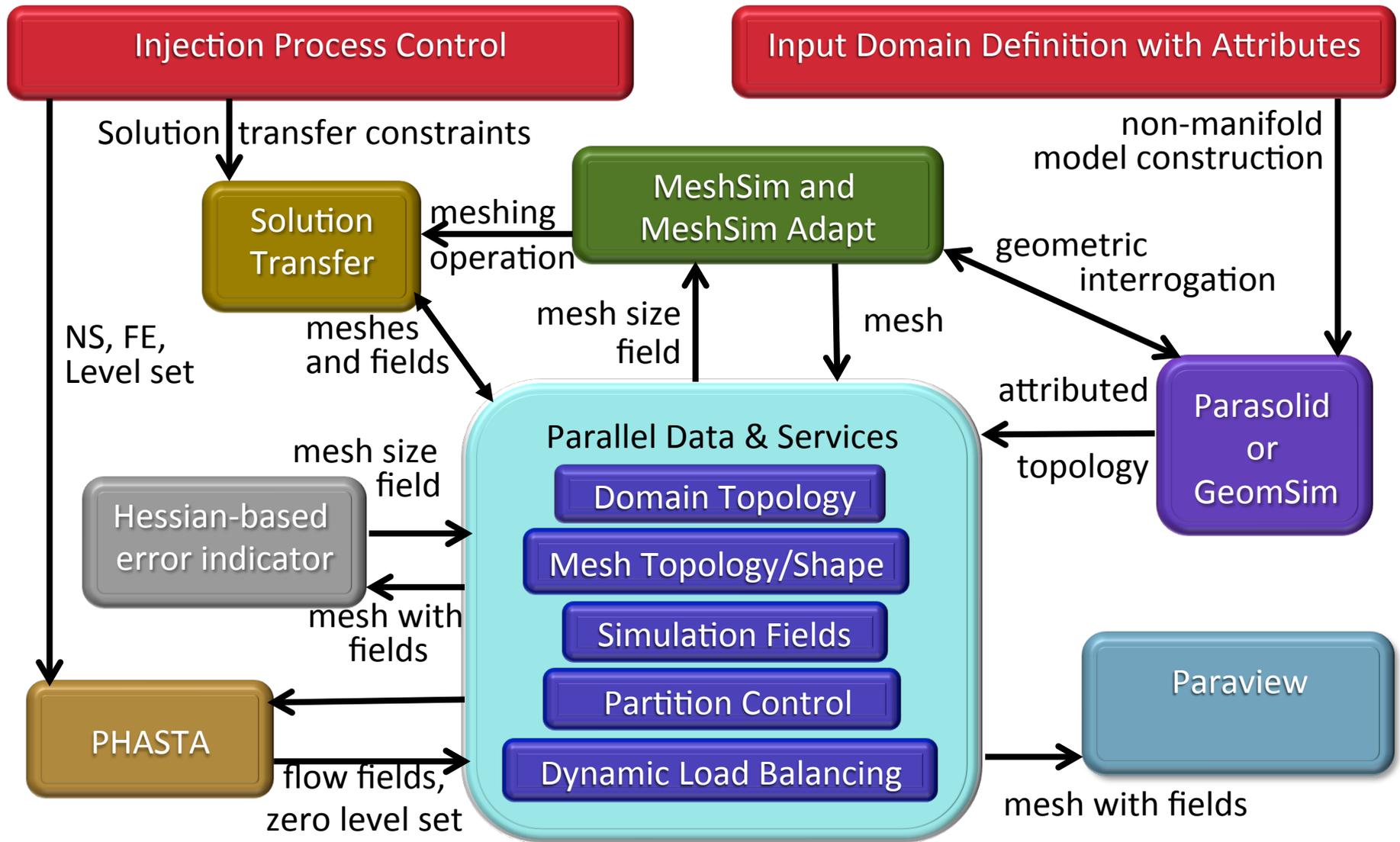
Case\AoA (L/D)	14 Up	14 Down
Baseline	10.2236	3.2286
Forced	10.6265	9.9921

15 m/s

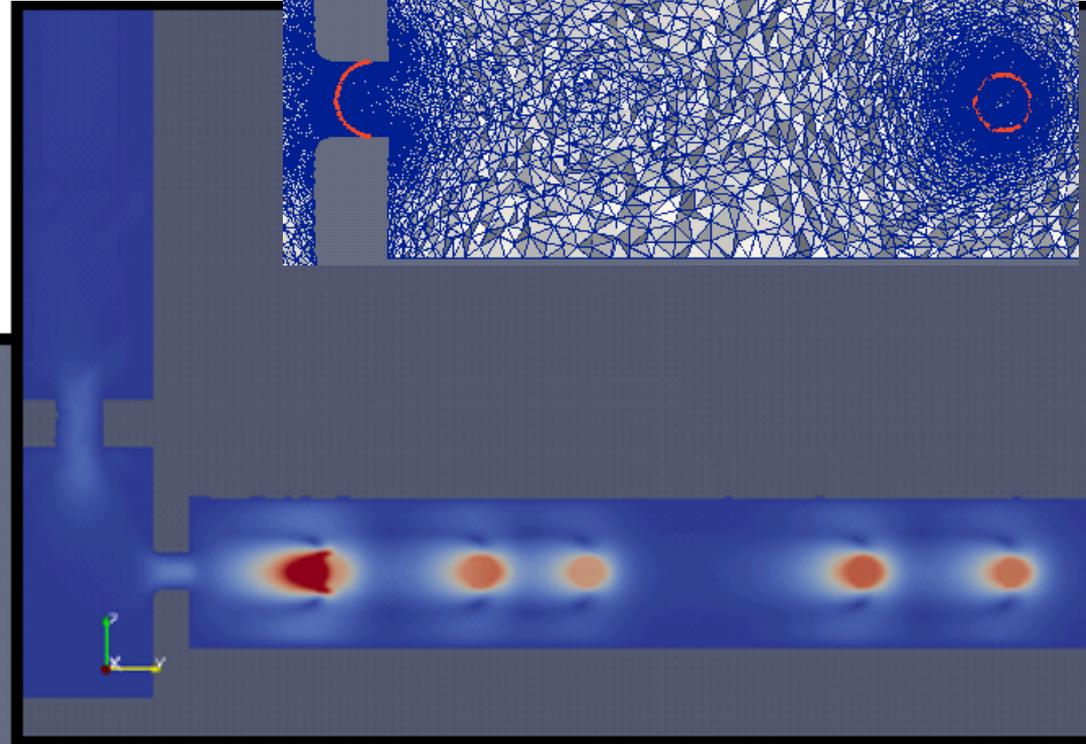
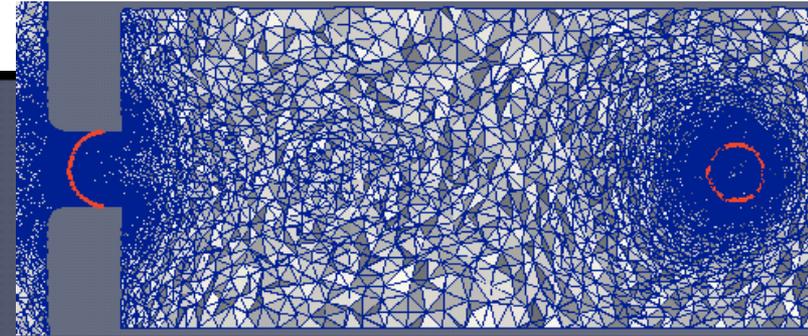


Leading-edge synthetic jets: 5 along span

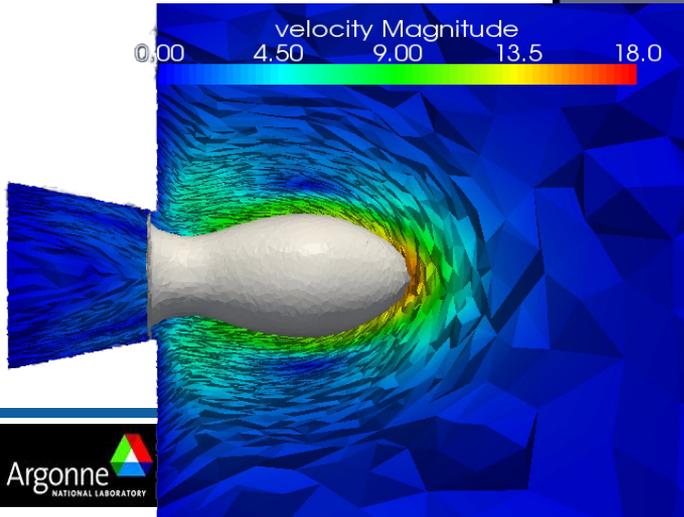


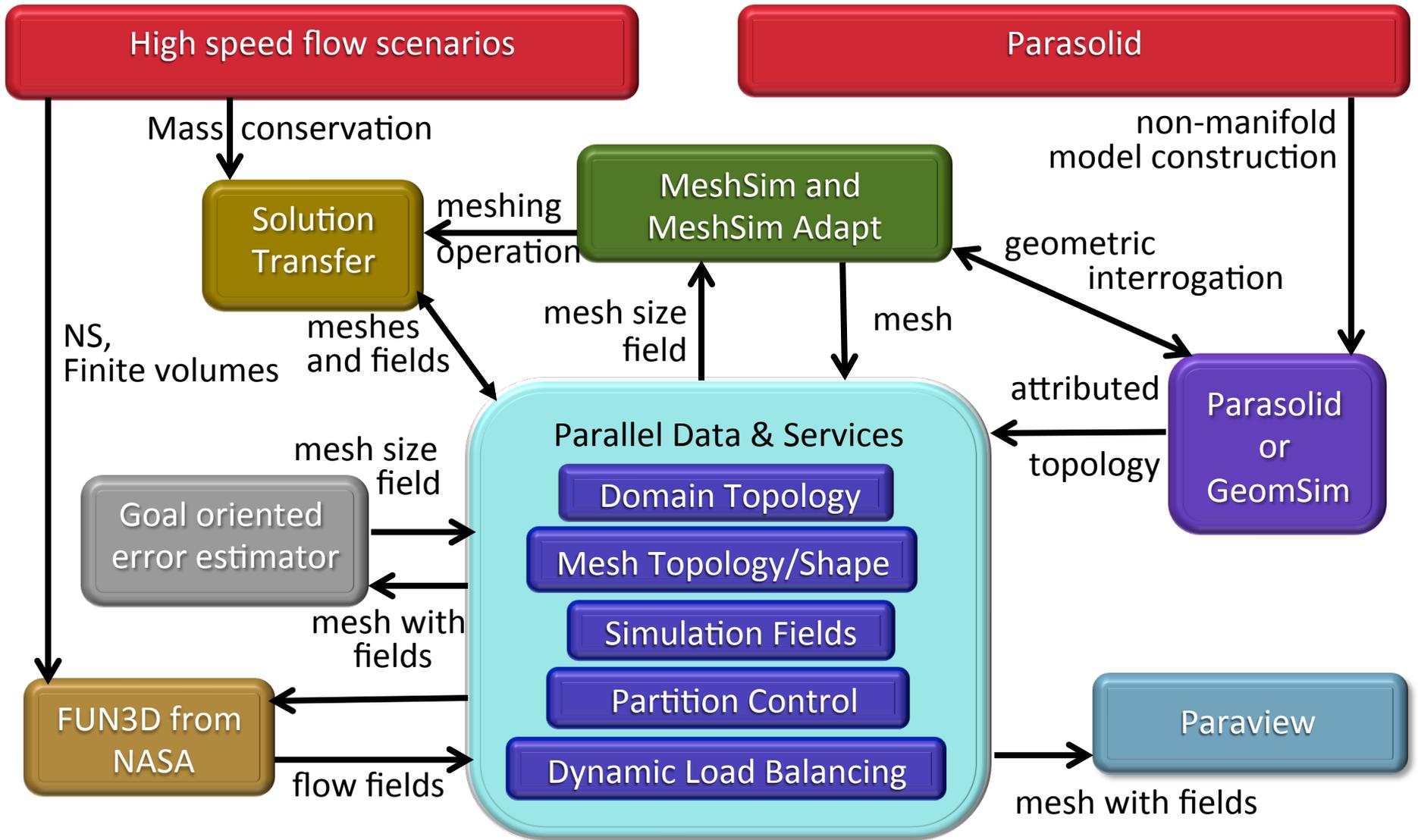


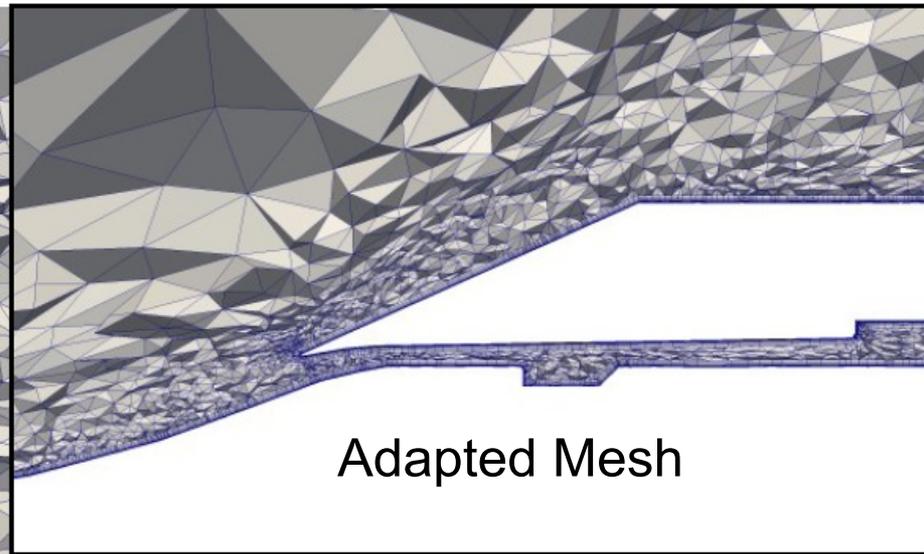
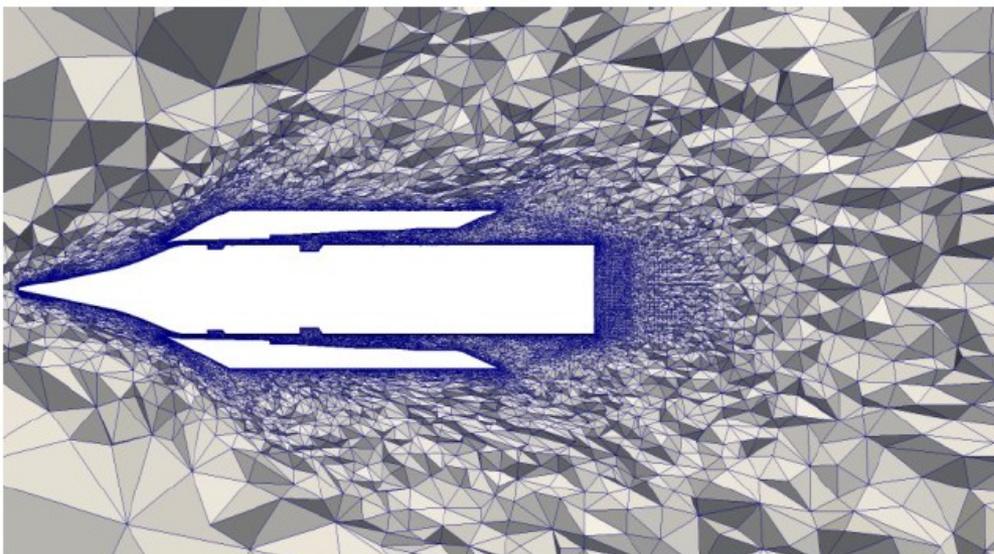
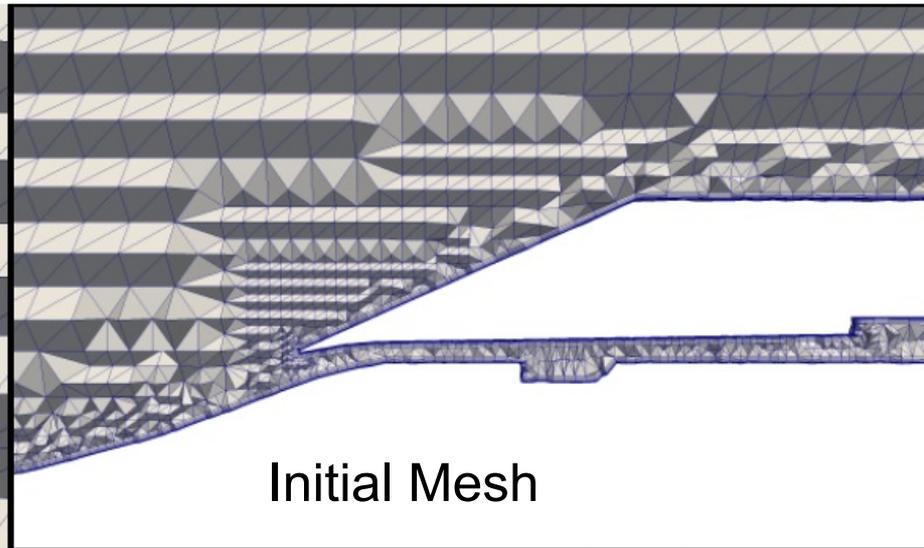
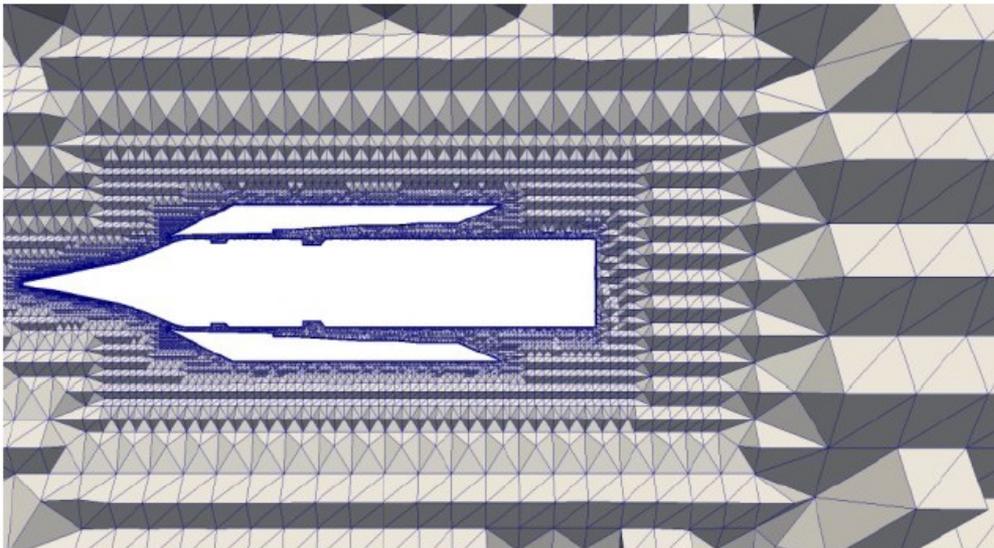
- Two-phase modeling using level-sets coupled to structural activation
- Adaptive mesh control – reduces mesh required from 20 million elements to 1 million elements

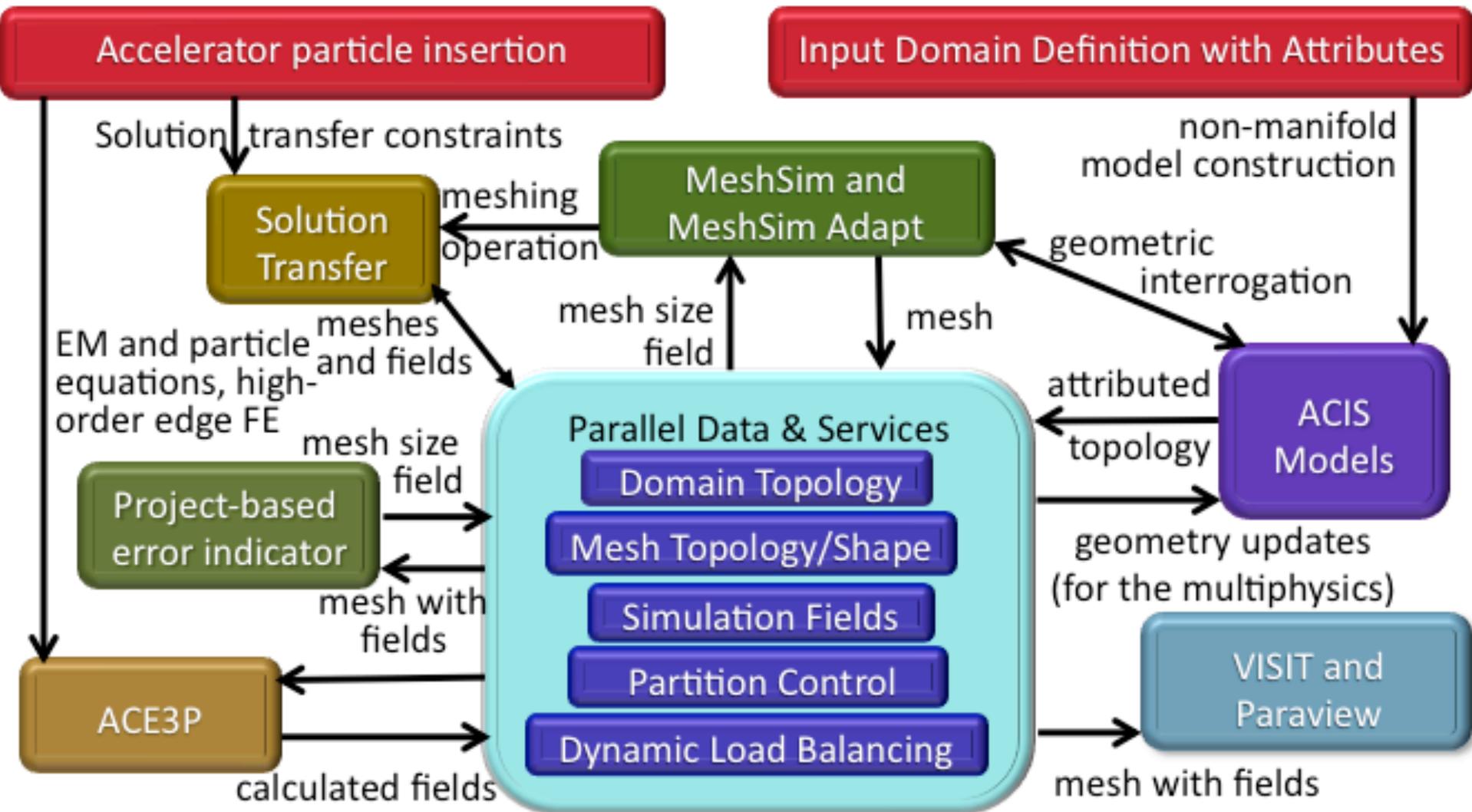


velocity Magnitude
0.00 4.50 9.00 13.5 18.0

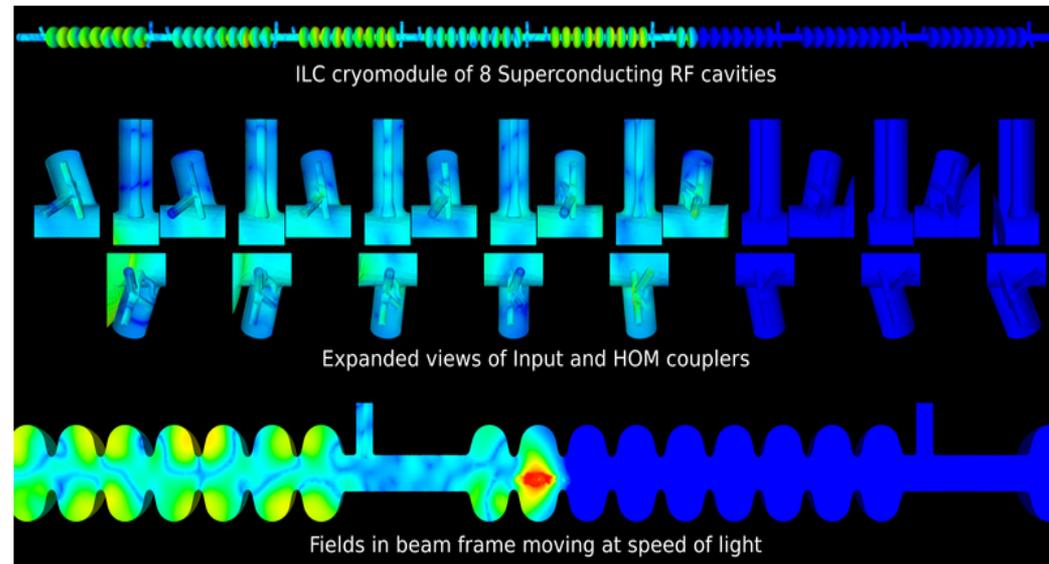
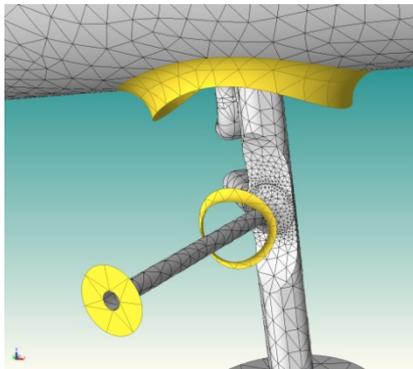
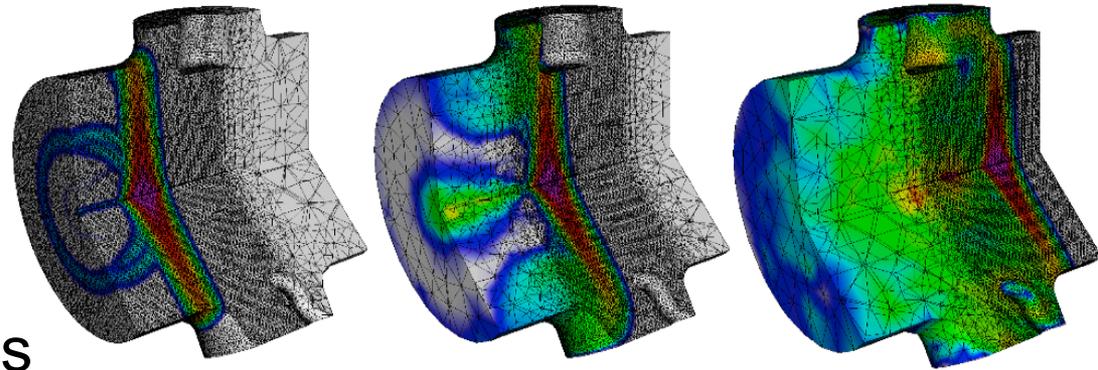


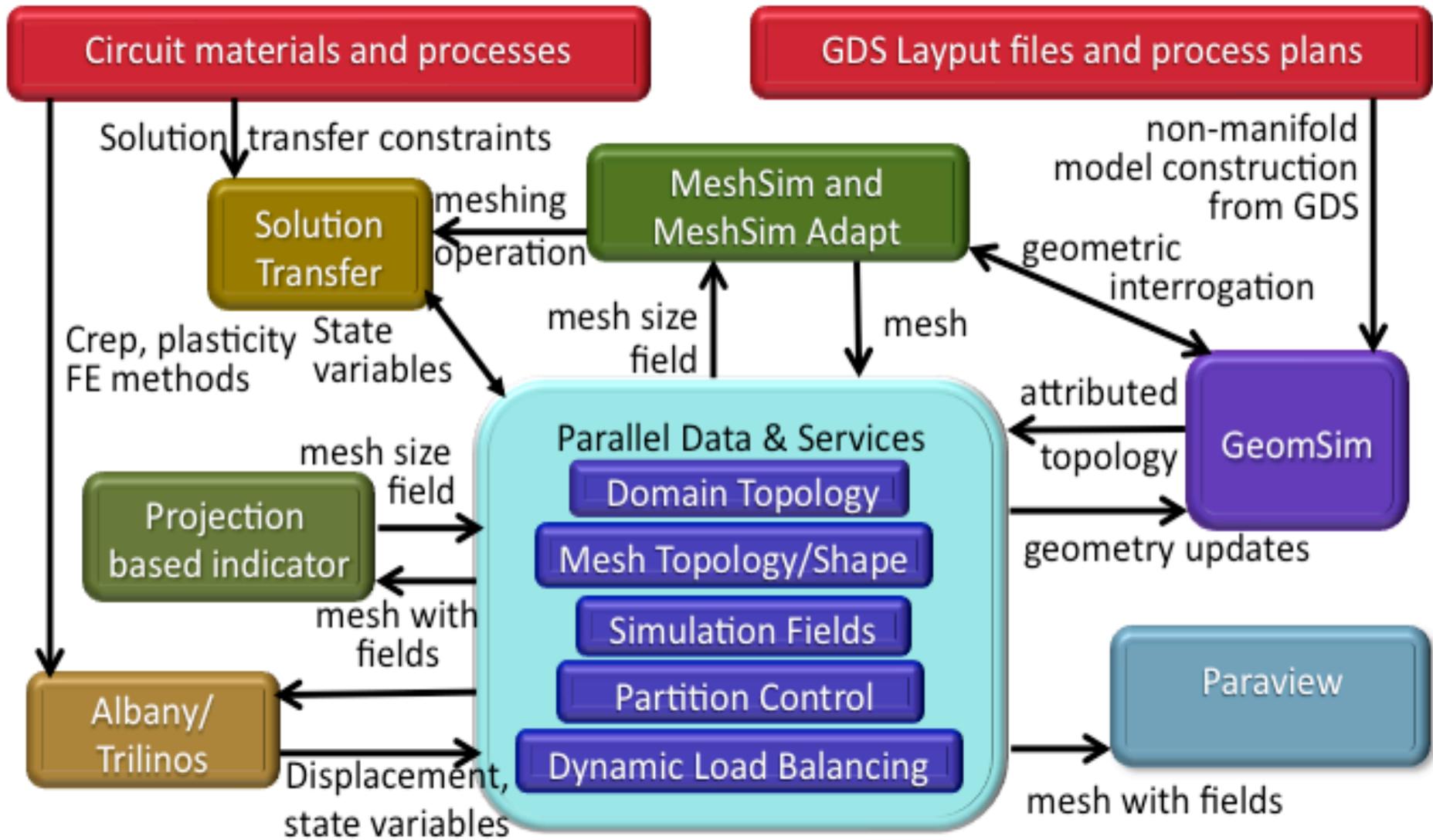






- Adaptation based on
 - Tracking particles
 - Discretization errors
- Full accelerator models
 - Approaching 100 cavities
 - Substantial internal structure
 - Meshes with several hundred million high-order curved elements

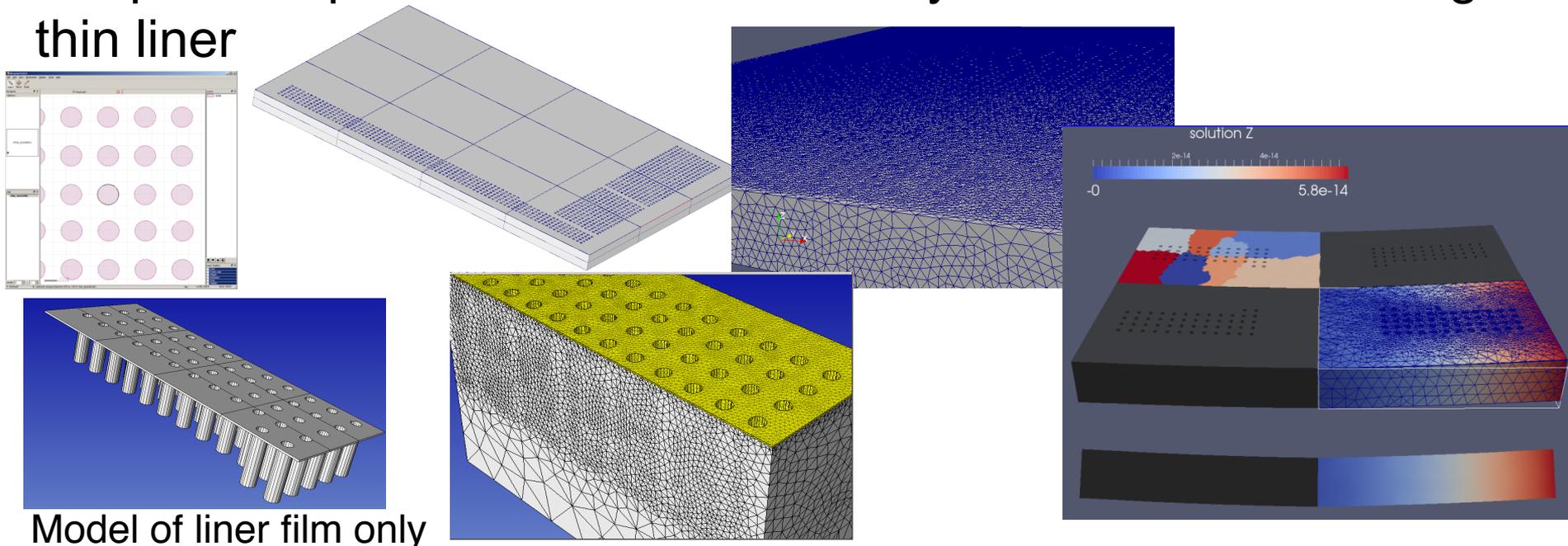




Must construct 3-D non-manifold solid from input geometry

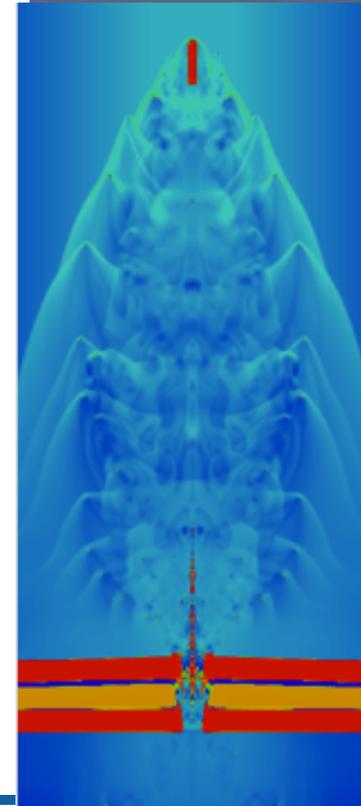
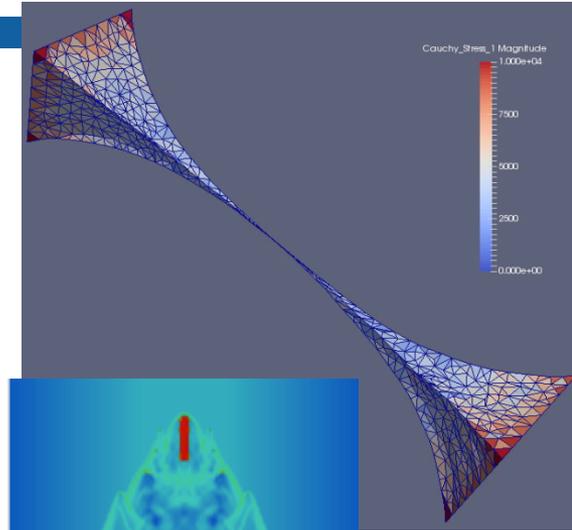
- Input domain defined in terms of 2-D layouts (gdsII/OASIS)
- Third dimension based on process knowledge
- A component has been developed to construct the model

Adaptive loop constructed for thermally loaded case including thin liner



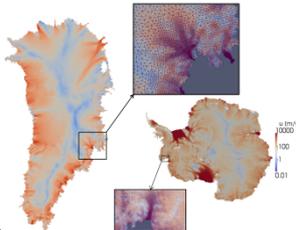
Combination of the FASTMath unstructured mesh technologies with agile multiphysics components and modules to develop:

- **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
- **Alexa**: a new approach at ferro-electric implicit-explicit (IMEX) shock multiphysics focused on *near*-Lagrangian methods enabled by unstructured adaptation

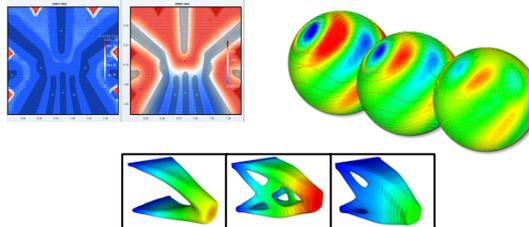


Application Impact: Ice Sheets

- 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

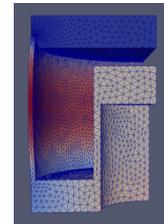
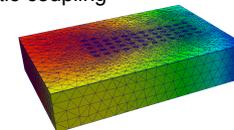


Additional Application Impact



Application Impact: Computational Mechanics

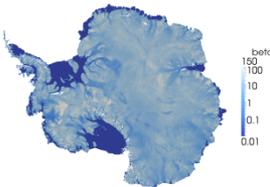
- Largest implicit problem solved in Albany to date: 1.7B degrees of freedom
- Initial capabilities for Schwarz multiscale coupling



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

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

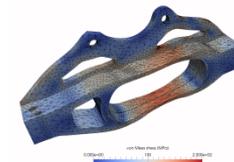
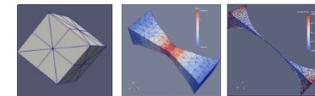


Application



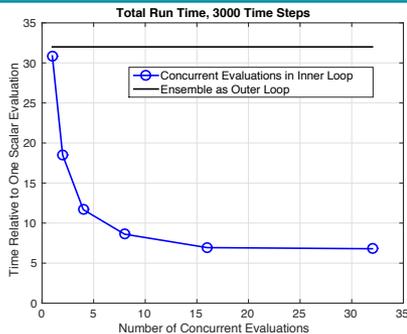
Mesh Adaptivity

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



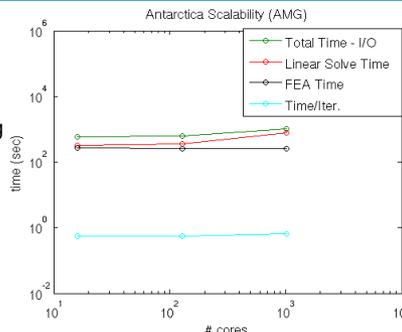
Embedded UQ

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



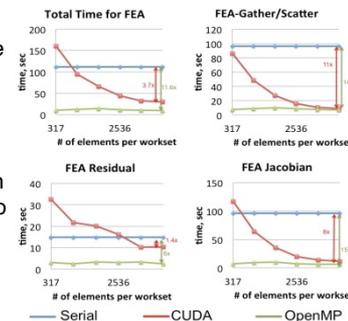
Scalable Linear Algebra

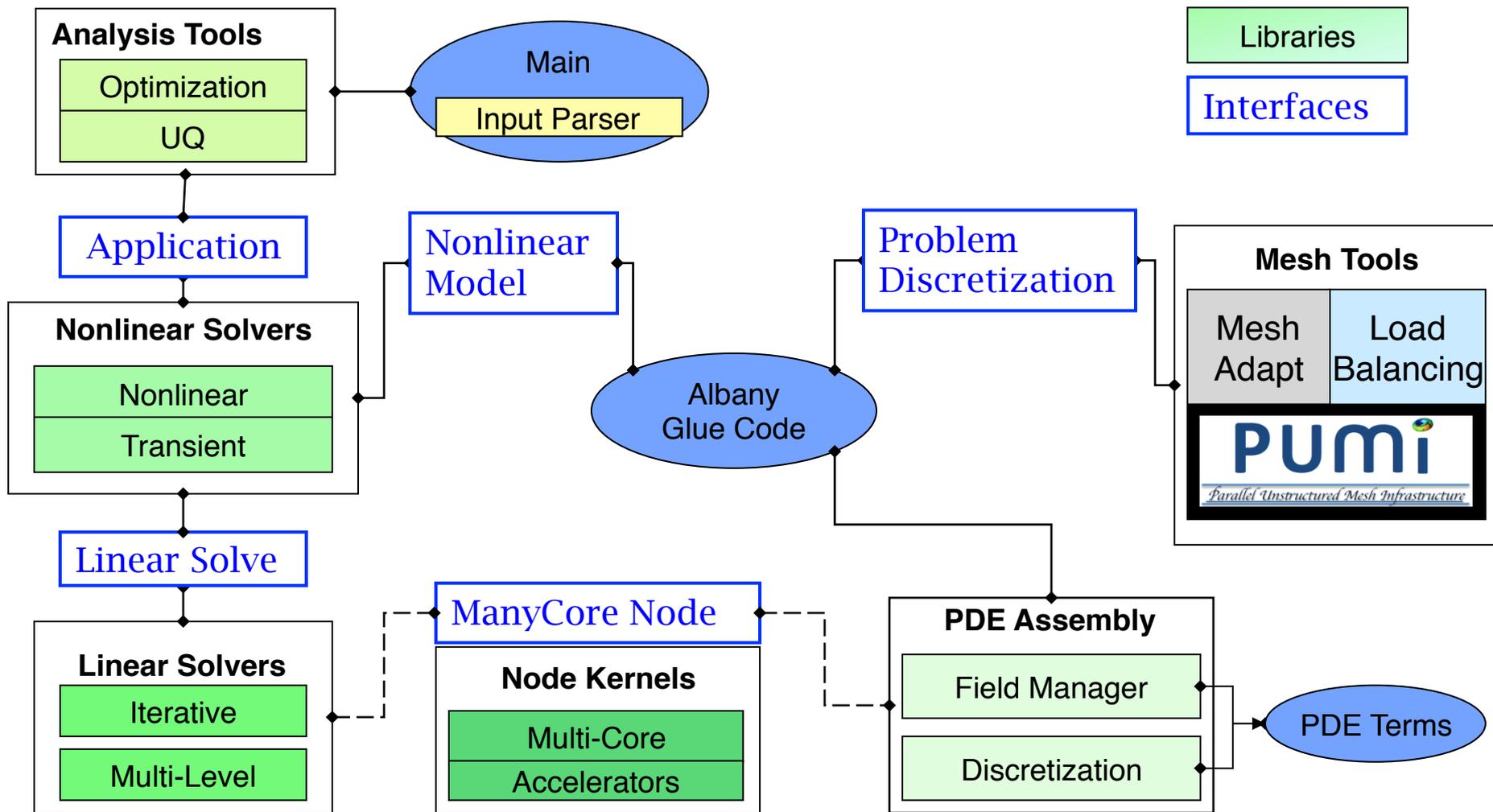
- Scalability of simulations requires effective preconditioning
- Multi-level solves are essential for the largest problems



Performance Portability

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

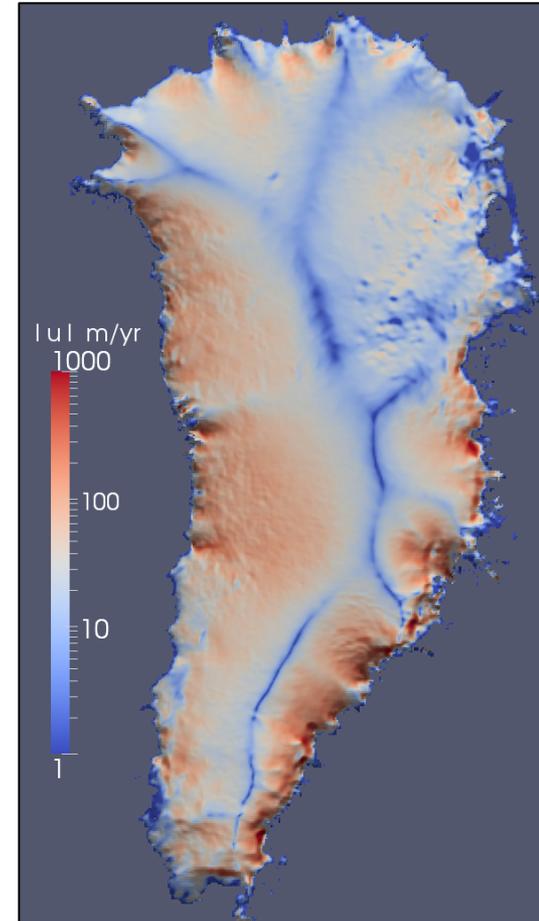
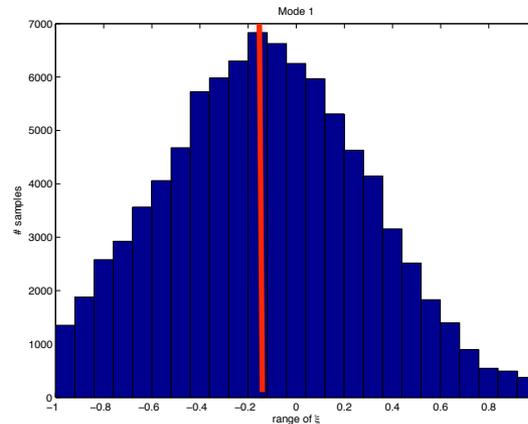




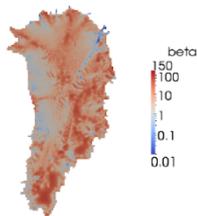
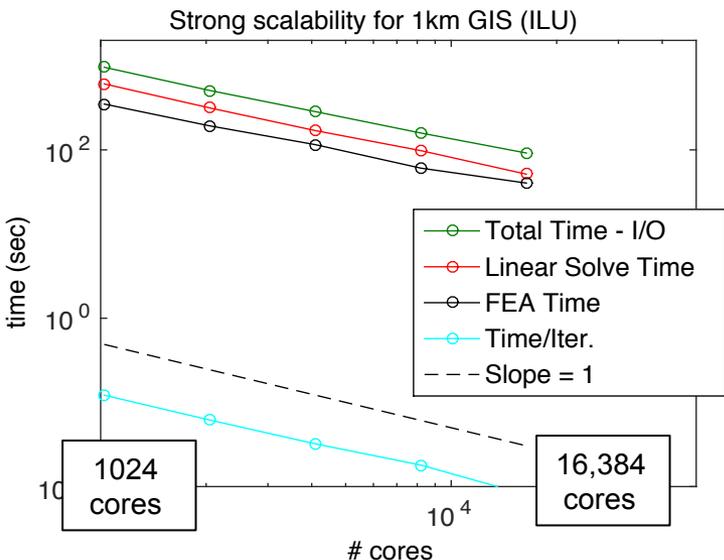
Develop and support a robust and scalable land ice solver based on the “First-Order” (FO) Stokes physics

- Uses dozens of libraries from Trilinos
- Scalable Multi-level linear solves (ML)
 - Convergence Study to 1.1B unknowns; 16K cores
 - Weak Scaling: 4096x size; 2.1x time
- Robust nonlinear solves (NOX)
- Same code base as PAALS adaptivity work

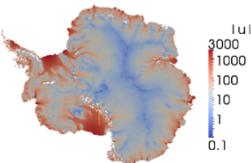
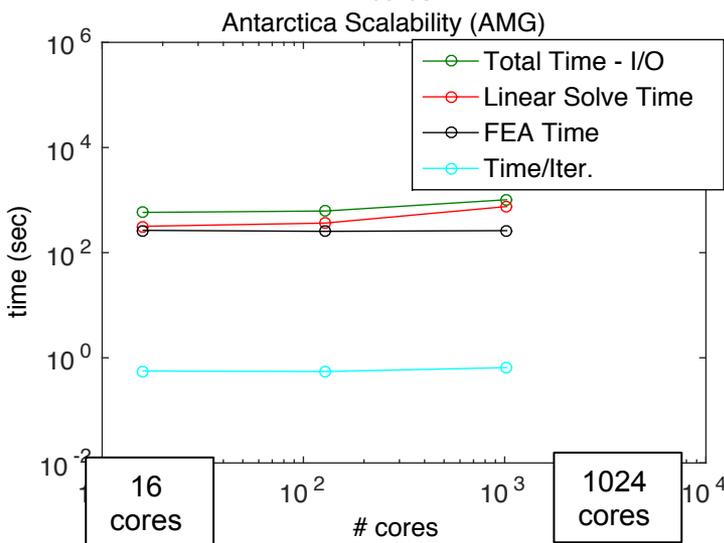
❖ Linked to Dakota for UQ and Calibration (KLE → PCE → MCMC)



Greenland Ice Sheet Surface Velocities



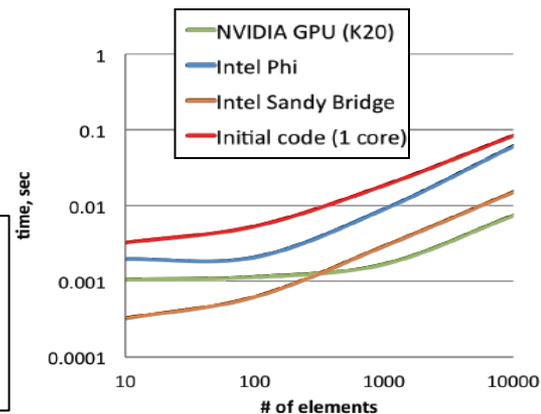
Strong scaling:
143M DOFs



Weak scaling:
8km, 5 layers to
2km, 20 layers

- Albany finite element assembly mini-app results suggests ~100 elements on the Phi and ~1000 element on the GPU is required to “break even”
- Table shows “speedup” using multicore and GPU in FELIX analysis

Note: Gather Coordinates routine requires copying data from host to GPU.



Kernel	Serial	16 Thds OpenMP	GPU
Viscosity Jacobian	20.39 s	2.06 s	0.54s
Basis Functions w/ FE Transforms	8.75 s	0.94 s	1.23s
Gather Coordinates	0.097 s	0.107 s	5.77s

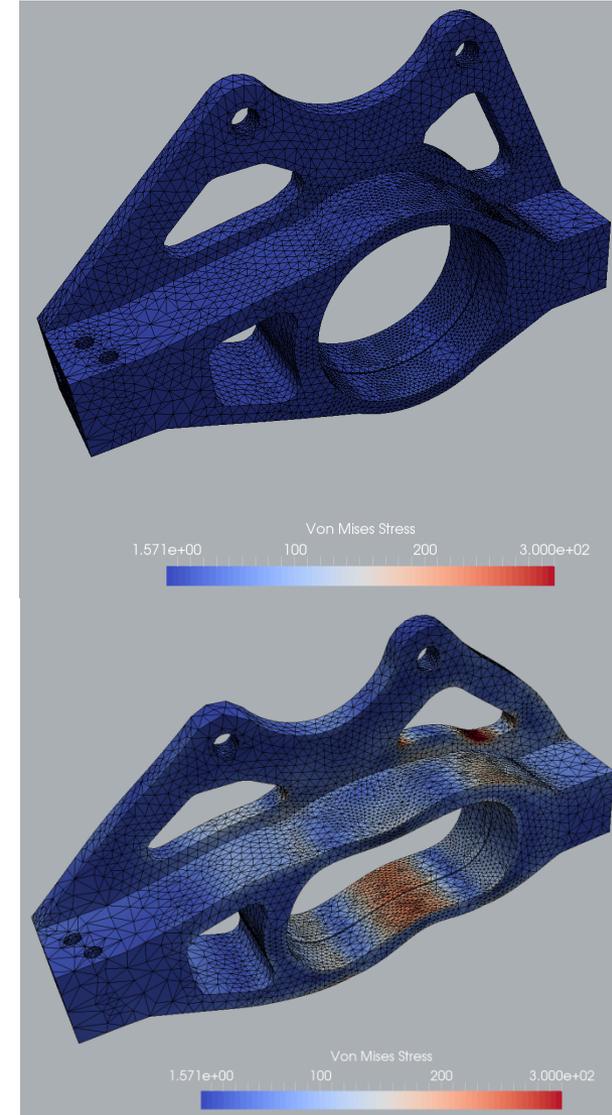
Courtesy of: Irina Tezaur (SNL)

Adaptive simulations of finite deformation plasticity with Albany

- Projects include modeling large deformations and weld failures

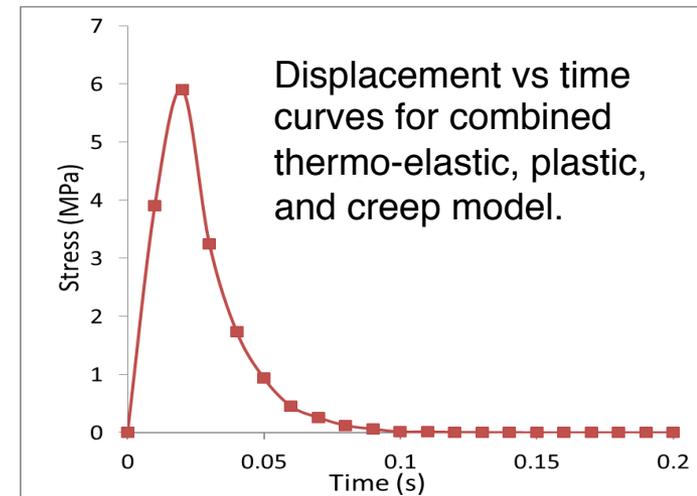
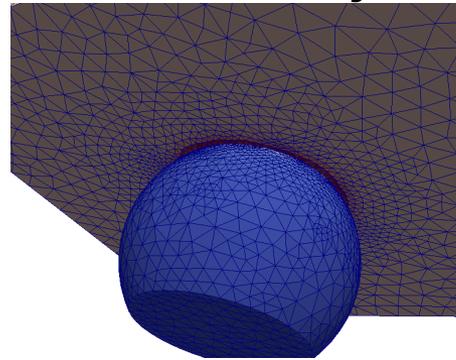
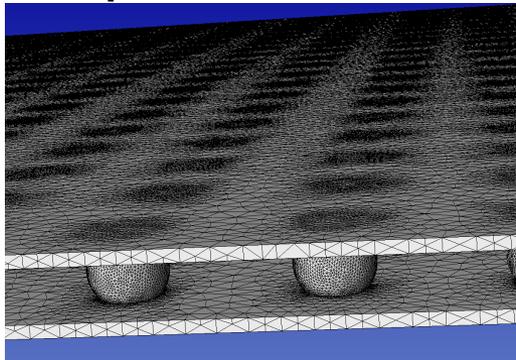
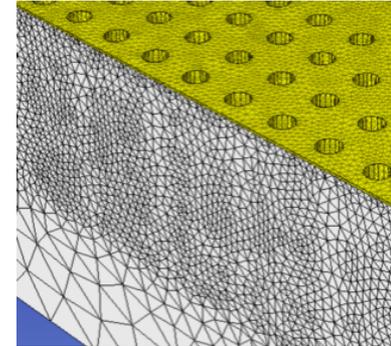
Efforts on adaptive loops that supports

- 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



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





FASTMath Unstructured Mesh Hand-On Session

Two tracks:

- Using SIGMA tools to construct mesh and its discretization for solving 3-D Laplacian
- Workflow demonstration using Simmetrix/PUMI/PAALS for parallel adaptive simulations

FASTMath SciDAC Institute



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

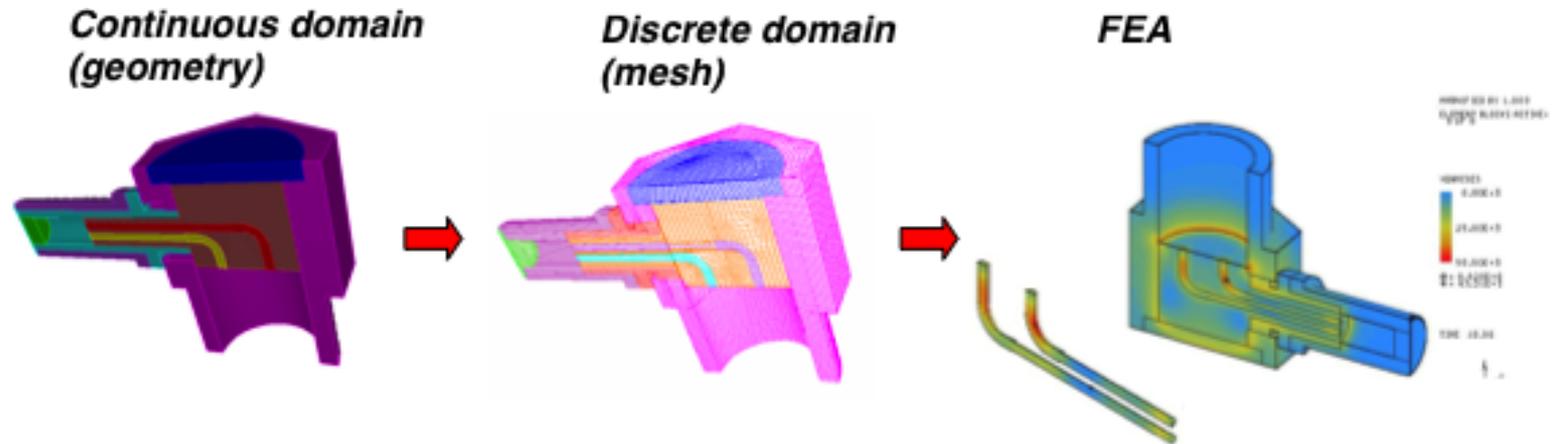
ATPESC 2015

Vijay Mahadevan and Iulian Grindeanu

FASTMath SciDAC Institute



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



- ⊙ [CGM](#) supports both open-source (OCC) and commercial (ACIS) geometry modeling engines.
- ⊙ [MOAB](#) provides scalable mesh (data) usage in applications through efficient array-based access; Support parallel I/O, 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 unstructured meshes with FEM.

- To utilize the SIGMA tools effectively, follow workflow to solve a simple 3-D Laplacian on an unit cube mesh.
 - ★ **Example 1: HelloParMOAB**
 - Introduction to some MOAB objects and load mesh from file
 - ★ **Example 2: LargeMesh**
 - Generate d -dimensional parallel mesh 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: GetEntities**
 - Query the parallel mesh to list the entities of various dimensions (elements, faces, edges, vertices)
 - Get entities and report non-vertex entity connectivity and vertex adjacencies.

★ Example 4: DMMoab Laplacian Solver

- Introduction to some DMMoab concepts
- Create DMMoab from file loaded
- Define field to be solved
- Setup linear operators and PETSc objects
- Solve linear operator
- Output and visualize

-
- Please consult the SIGMA website for help on examples.

<http://sigma.mcs.anl.gov/sigma/atpesc-2015>

- All other MOAB questions can be directed to

moab-dev@mcs.anl.gov



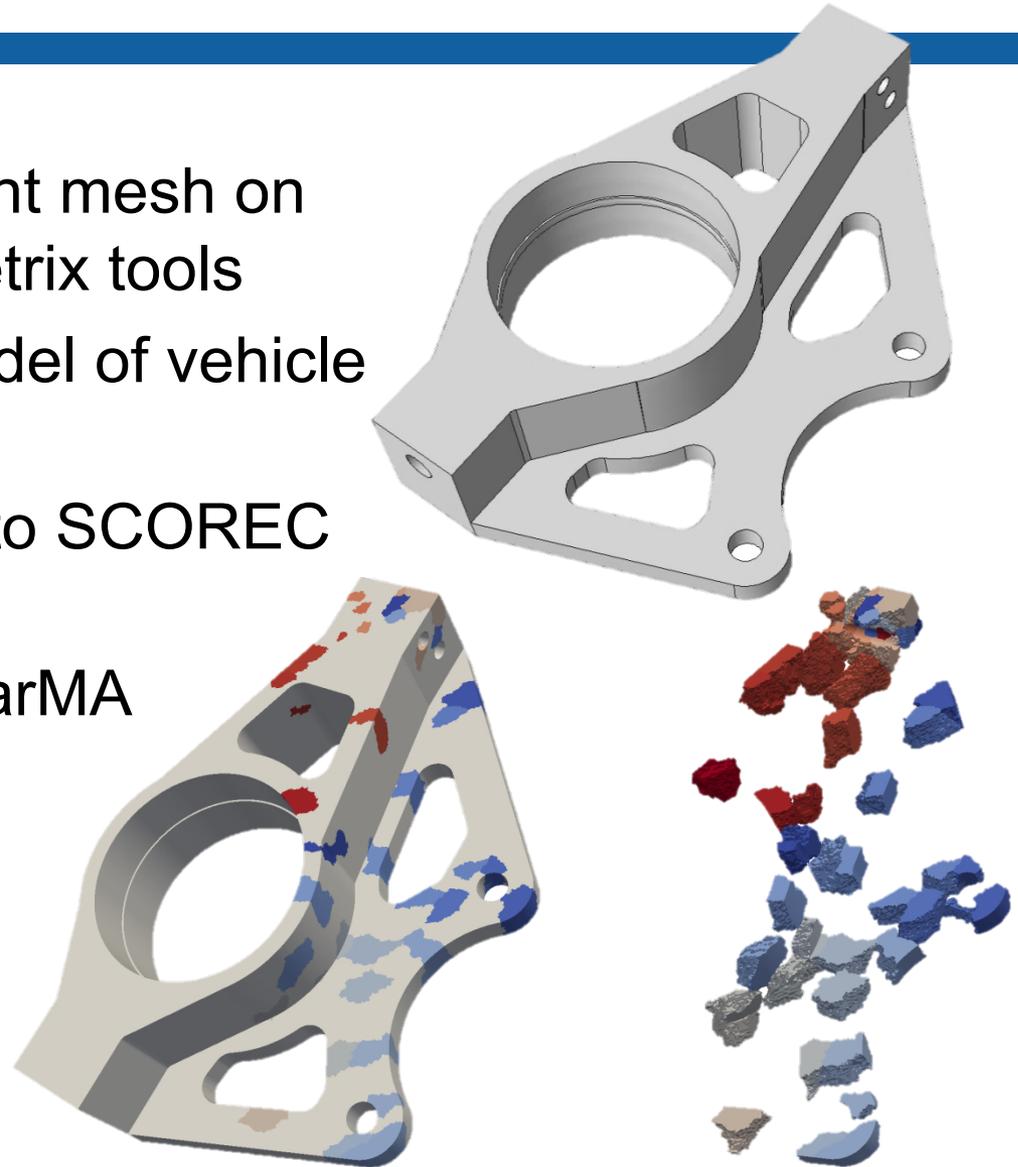
Workflow demonstration using Simmetrix/PUMI/ PAALS for parallel adaptive simulations

Presenters: Cameron W. Smith and Glen Hansen

FASTMath SciDAC Institute



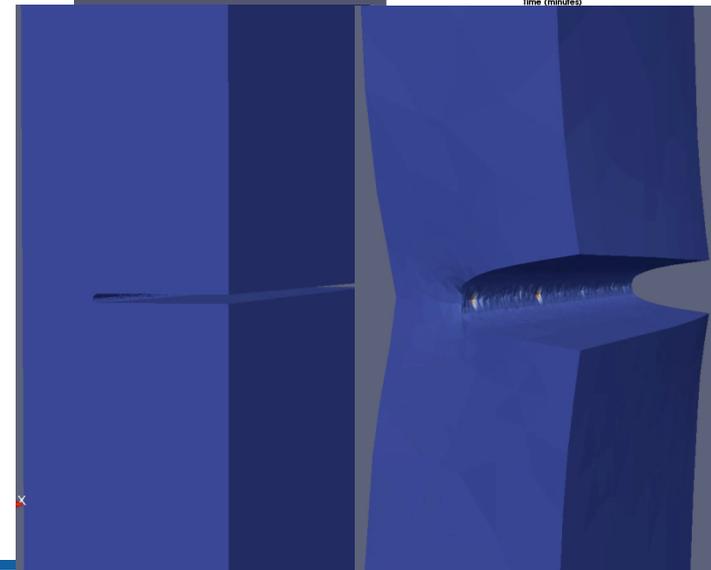
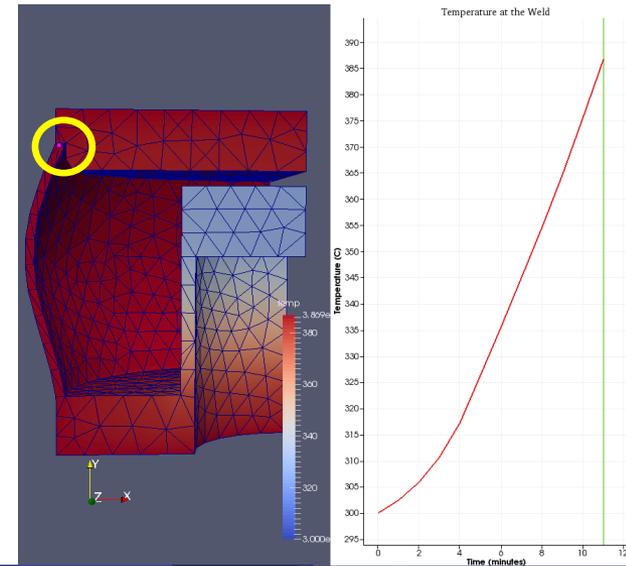
- Parallel Mesh Generation
 - Generate a 13M element mesh on 128 cores using Simmetrix tools
 - Complex geometric model of vehicle suspension upright
 - In-memory conversion to SCOREC mesh data structures
- Partition via Zoltan and ParMA
 - Recursive inertial bisection method
 - ParMA Vtx>Elm
 - Partition on 512 cores



■ PAALS

- In-memory parallel adaptive loop to analyze thermo-mechanical breach
- Running on 1024 cores
- Combining
 - Parallel Mesh Adapt
 - Quadratic mesh elements
 - SPR based error estimation
 - Local solution transfer of history dependent state variables
 - Predictive load balancing

■ Visualization with ParaView



■ Hands-on exercise

- <https://github.com/gahansen/Albany/wiki/PAALS-Tutorial>

■ 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



■ 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]