FASTMath Unstructured Mesh Technologies

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

FASTMath SciDAC Institute

Unstructured Mesh Technology and Application Team

Argonne National Laboratory

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

Lawrence Livermore National Lab.

- Barna Bihari
- *Lori Diachin*

Sandia National Laboratories

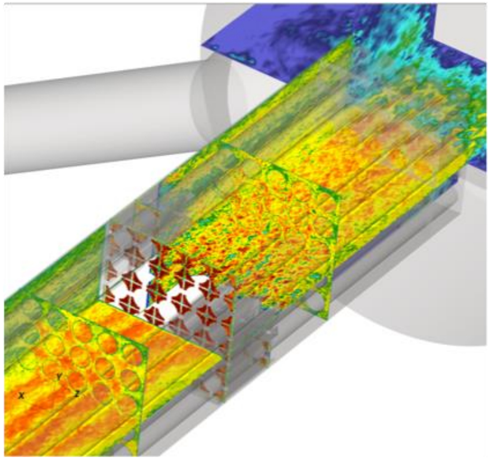
- Karen Devine
- *Glen Hansen*
- Vitus Leung
- Jake Ostien

Rensselaer Polytechnic Inst.

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

Presentation Outline

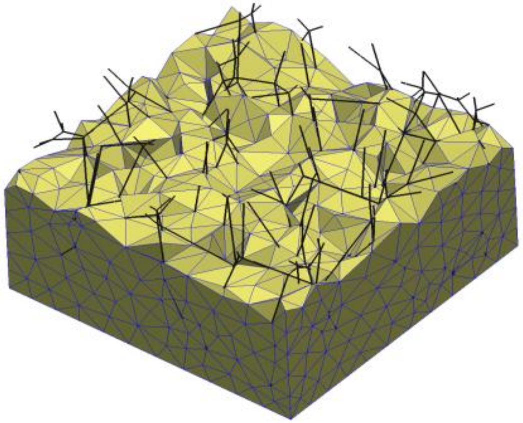
- Unstructured mesh methods and the need for unstructured mesh components for use by analysis code developers
- Core unstructured mesh components:
  - Parallel Mesh infrastructures
  - Mesh Generation, Adaptation, Optimization
  - Fields
  - Solution transfer
- Dynamic load balancing
- Unstructured mesh/solver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



Unstructured Mesh Methods

Advantages of unstructured mesh methods

- Easily applied to general geometries
  - Fully automated procedures to go from CAD to valid mesh
- Can provide highly effective solutions
  - Easily fitted to geometric features
  - Easily graded
  - General mesh anisotropy to account for anisotropic physics possible



- Given a complete geometry (e.g., CAD solid model), 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

- Require the use of more complex data structures to describe
  - More complex to program, particularly in parallel
- Although they can give the highest accuracy on a per degree of freedom basis that takes specific care and effort
  - The quality of element shapes influences solution accuracy – the degree to which this happens a function of the discretization method
  - Poorly shaped element 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

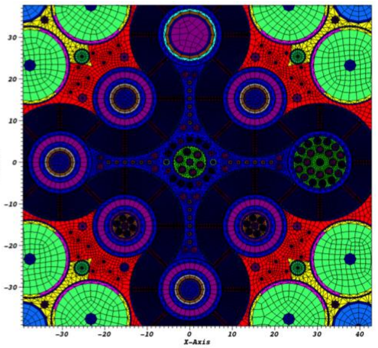
Unstructured Mesh Methods

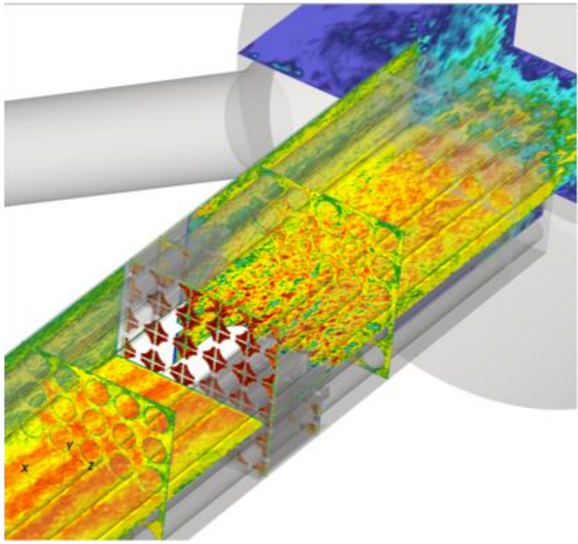
Goal of FASTMath unstructured mesh developments include:

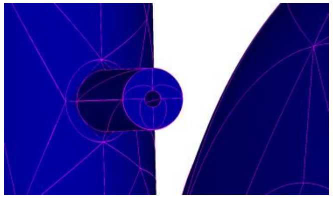
- Provide component-based tools that support analysis code developers and users to take full advantage of unstructured mesh methods
- Develop those components to operate through multi-level APIs that increase interoperability and ease of integration
- Address technical gaps by developing specific unstructured 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 first wall chemistry & dynamics (XOLOTL)
- Fusion Plasmas (M3DC1)
- High-order CFD on (Nektar++)
- High-speed viscous flows (FUN3D)
- Monte Carlo neutron transport (DAG-MCNP)
- Mortar element Structural Mechanics (Diablo)
- Multiphase reactor flows (PHASTA)
- SEM-based CFD (Nek5000)
- Solid Mechanics (Albany)



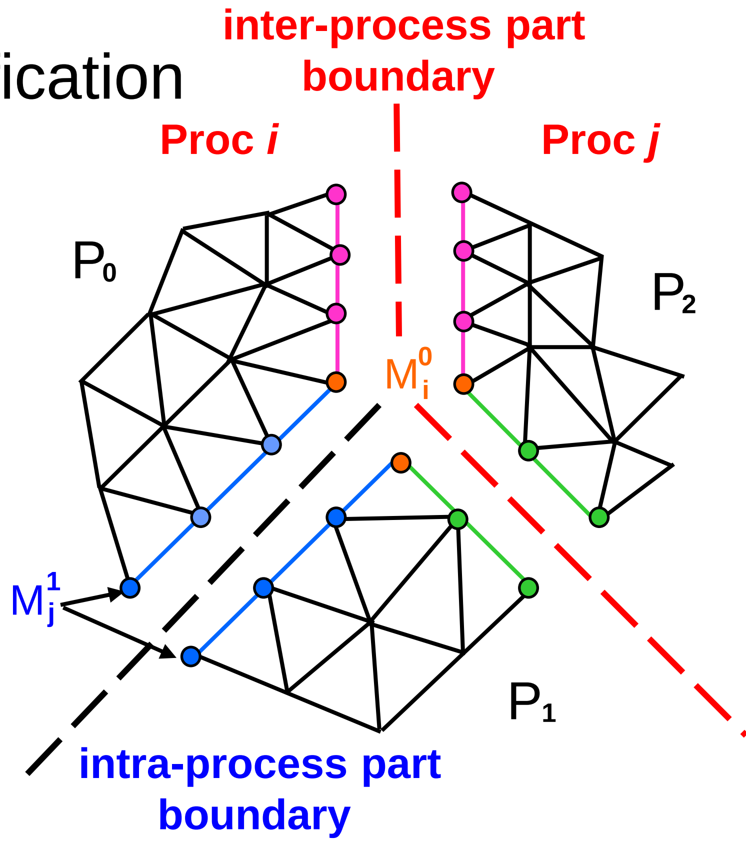




Parallel Mesh Infrastructure

Key unstructured mesh technology needed by applications

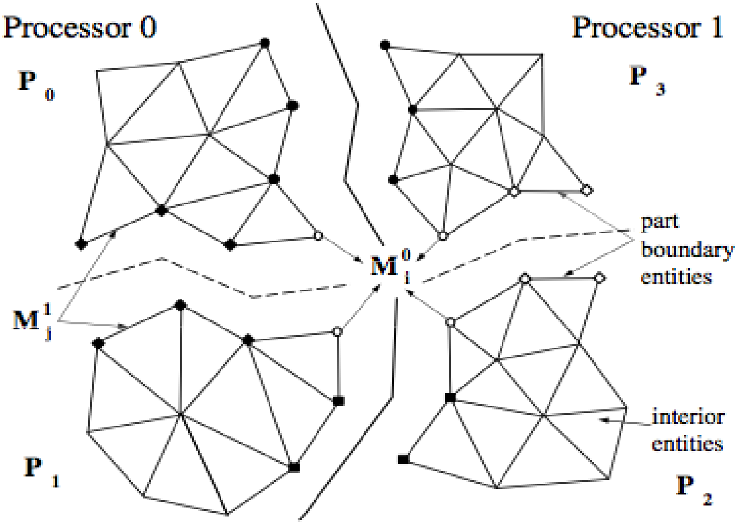
- Effective parallel mesh representation
- Base parallel functions
  - Partitioned mesh control and modification
  - Read only copies of types needed
  - Associated data, grouping, etc.
- Key services
  - Load balancing
  - Mesh-to-mesh solution transfer
  - Mesh optimization and adaptation
- Two FASTMath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



Parallel Mesh Based on Partitioning Mesh

Mesh part is a set of mesh entities assigned to unique part

- Treated as a serial mesh with part boundaries
- Part boundaries maintain links to neighboring mesh entities

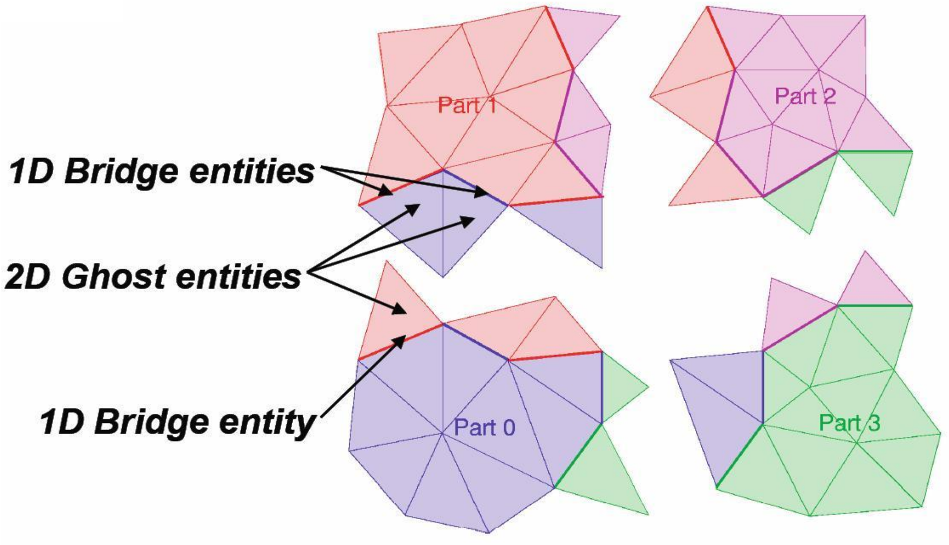


Mesh Migration

- 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

Localizing 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 be able to use a posteriori information to improve mesh
- Must be able to account for original geometric domain
- Want general, and specific, anisotropic adaptation capabilities

Mesh Shape Optimization

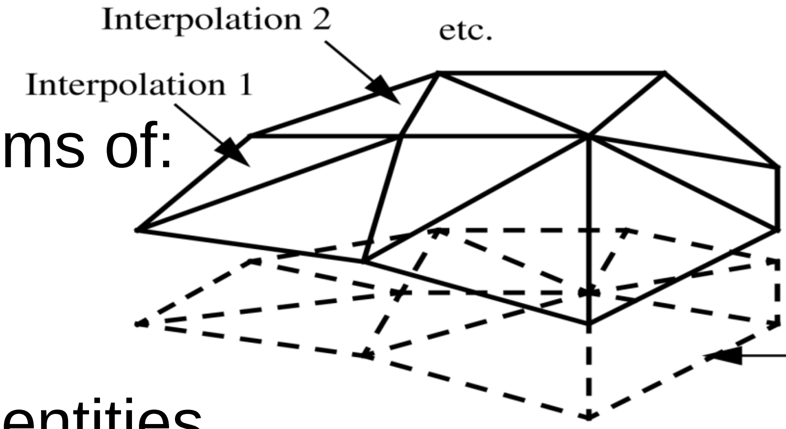
- Need to control element shapes as needed by the various discretization methods for maintaining accuracy and efficiency

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.



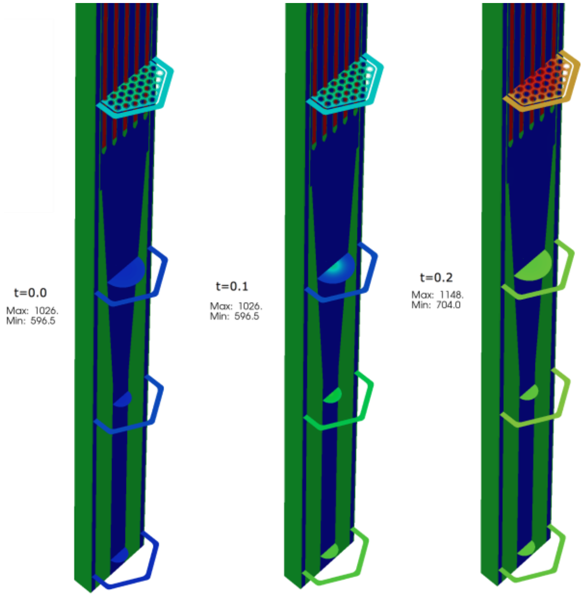
SIGMA Unstructured Mesh Infrastructure

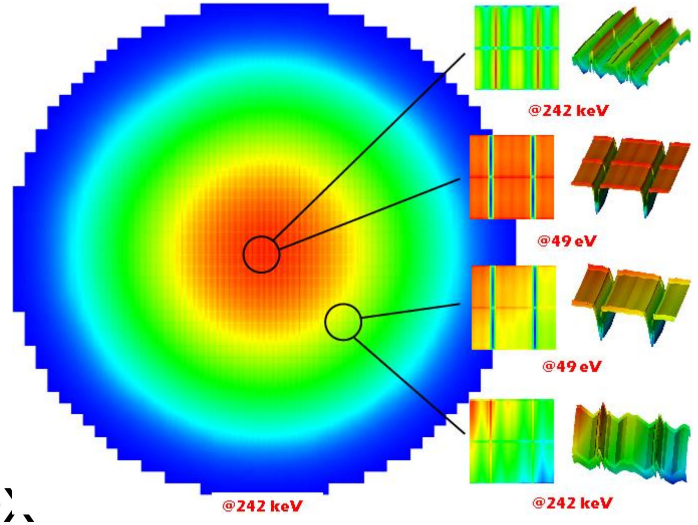
SIGMA(CGM/MOAB/Lasso/MeshKit) (http://sigma.mcs.anl.gov)

- CGM supports both open-source (OCC) and commercial (ACIS) geometry modeling engines.
- MOAB provides scalable mesh (data) usage in applications on >32K cores through efficient array-based access with support for parallel HDF5 I/O format and in-situ visualization.
- MeshKit provides unified meshing interfaces to advanced algorithms and to external packages (Cubit/Netgen).
- Components for discretization and mesh-to-mesh coupling
  - CouPE provides component based multi-physics solver capability utilizing underlying scalable solution transfers.
  - PETSc – MOAB interface simplifies discretization of PDE on unstructured meshes and solution through PETSc infrastructure.

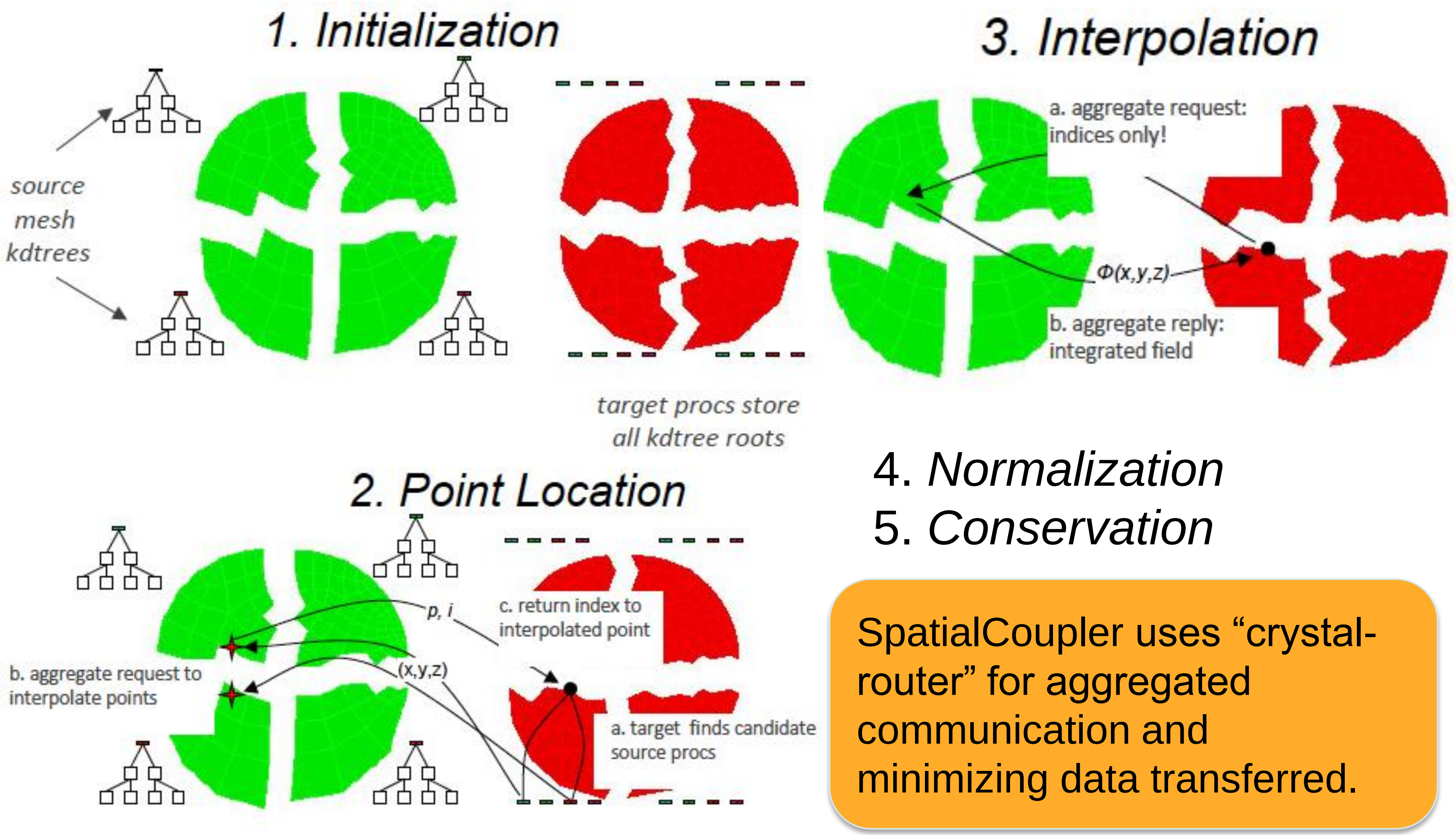
Parallel Point Location

- Goals: Simplify geometry search and unify discretization kernels with a flexible interface
- Geometry search: support uniform 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
- Leverage scalable mesh (MOAB) and solver (PETSc) structures to build a framework (SHARP) to perform complex nuclear reactor analysis problems.





Parallel Point Location

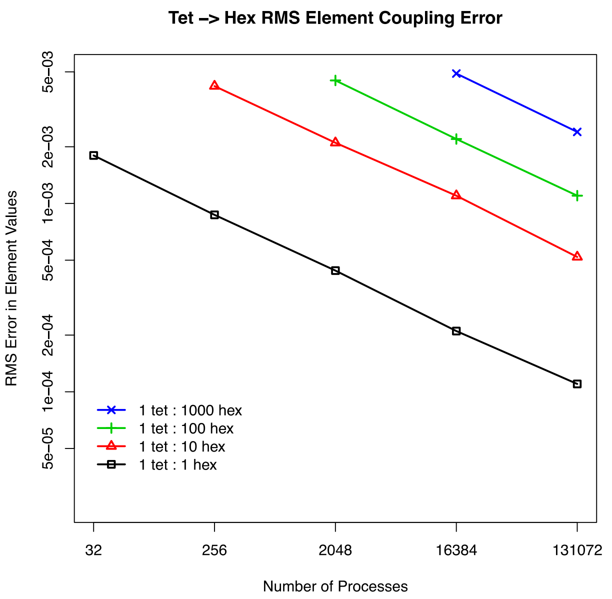


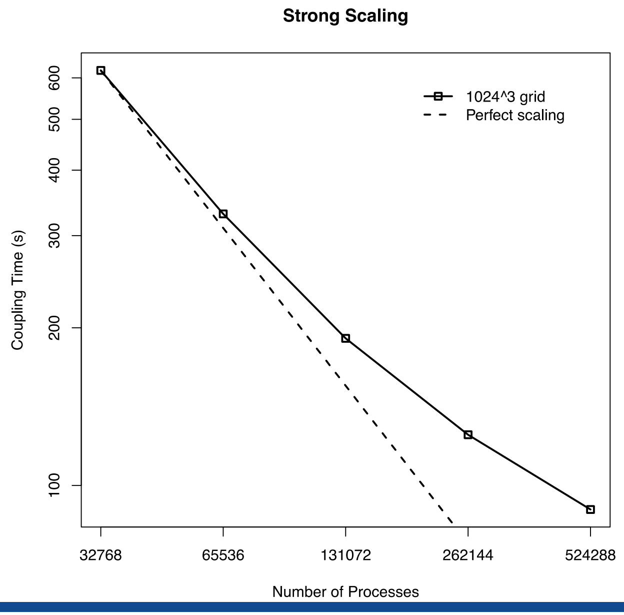
4. *Normalization*
5. *Conservation*

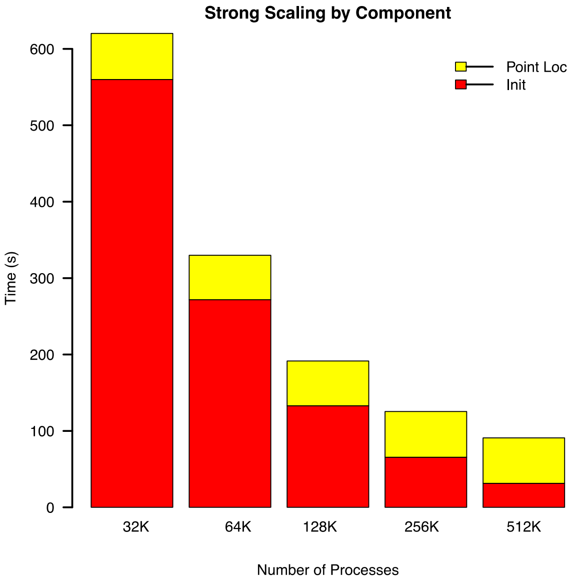
SpatialCoupler uses “crystal-router” for aggregated communication and minimizing data transferred.

Solution Transfer Scalability

- Demonstrated 70% strong scalability of the solution transfer implementation in MOAB up to 512K cores on BG/Q.
- Points/rank = [2K, 32K] (averaged around 10K).
- Initialization costs amortized over multiple interpolations!
- Bottleneck: Kd-tree scales as O(nlog(n)); Consider BVH/BIH trees to attain O(log(n)) time complexity.


Tet –> Hex RMS Element Coupling Error


Strong Scaling

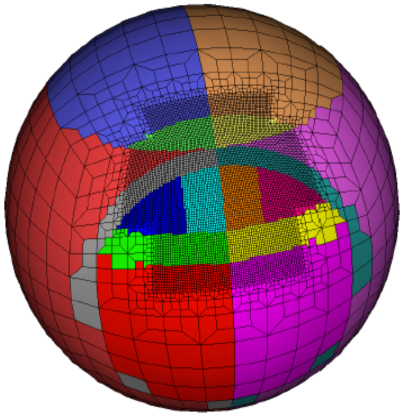

Strong Scaling by Component

Mesh Intersections + Dynamic partitioning

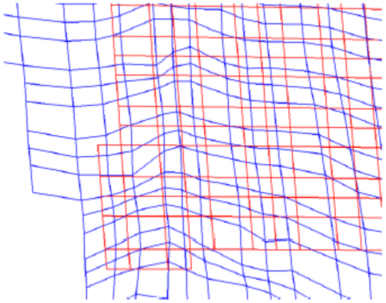
- MOAB parallel infrastructure and dynamic moving mesh intersection algorithm used to track multi-tracer transport.
- Implemented a scalable (linear complexity), conservative, 2-D remapping algorithm.
- Efficient and balanced re-distribution of meshes implemented internally with Zoltan interfaces

HOMME

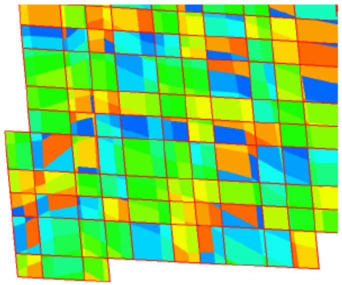
Global Partitioned Grid



Euler (Red), Backtraced (Blue) grids



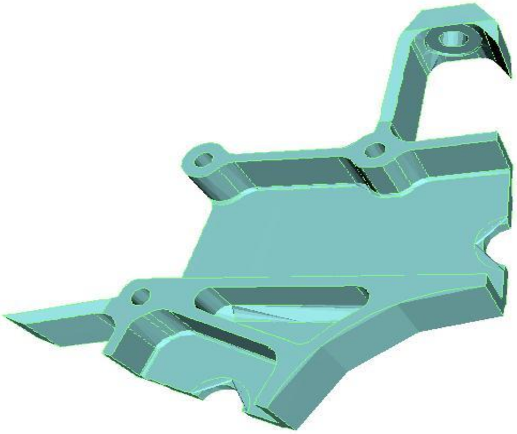
Intersections grouped by Euler cells



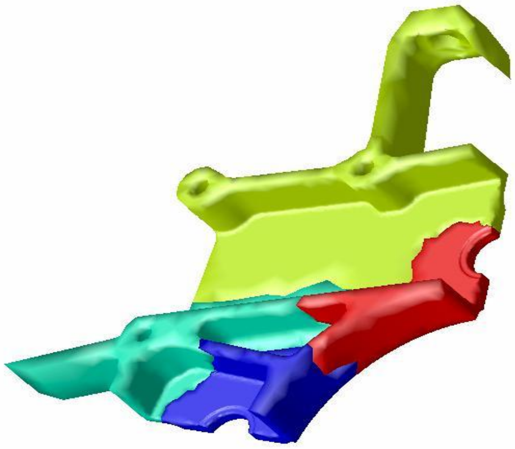
- Collaborative effort (ACES4BGC + FASTMath + SUPER)

PUMI Unstructured Mesh Infrastructure

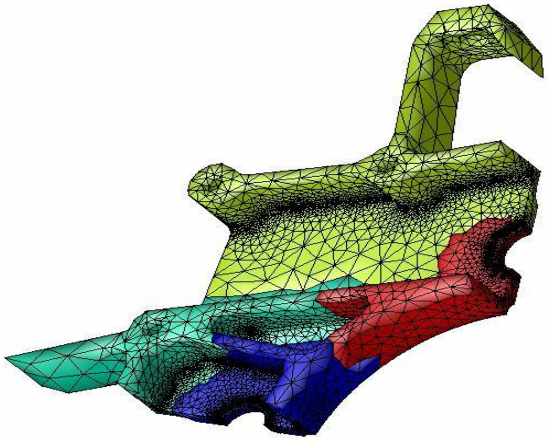
- Employs a complete mesh representation to provide any adjacency in O(1) time
- Parallel control through partition model that supports
  - All interprocess communications
  - Effective migration of mesh entities
  - Generalized read only copies



Geometric model



Partition model

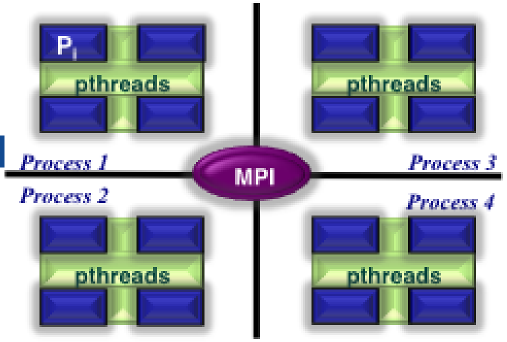


Distributed mesh

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 from CAD
  - 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 meshes with 92 billion elements on ¾ million cores

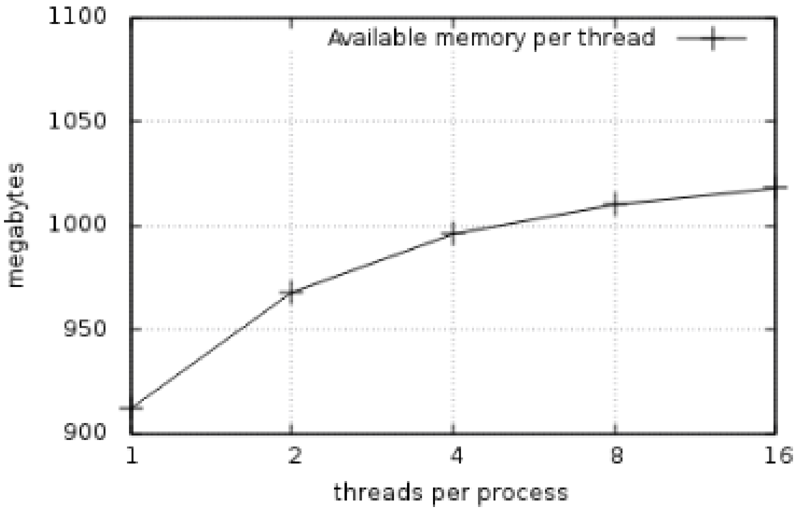
Architecture Aware PUMI



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

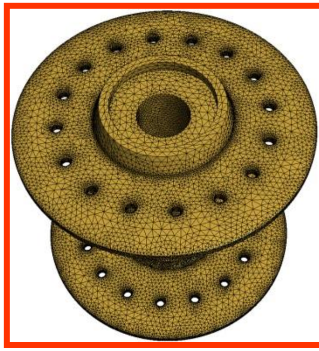


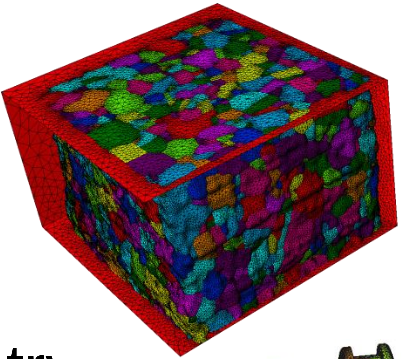
Use of Intel Phi accelerators

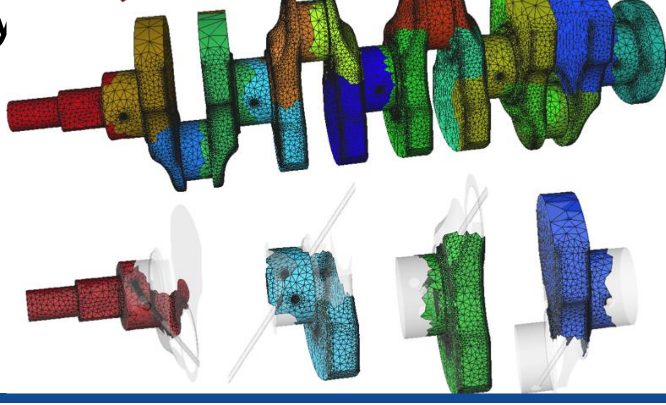
- On an equal number of Phi and BG/Q nodes
  - 1024 → 2048 partitioning is 40% faster on Stampede
  - 2048 → 4096 partitioning 8% slower on Stampede

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





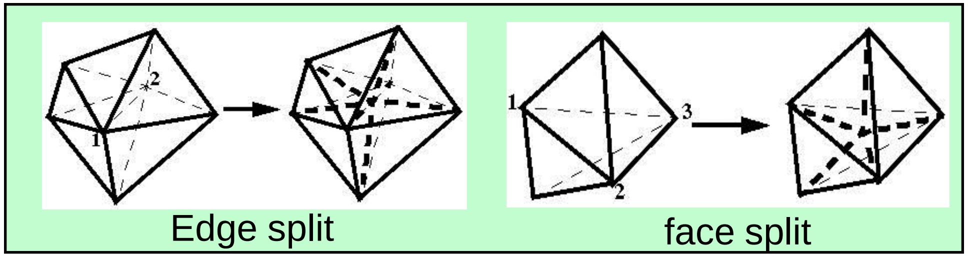


General Mesh Modification for Mesh Adaptation

- 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)

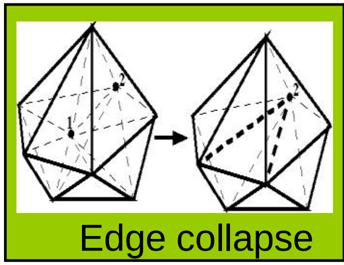
Mesh Adaptation by Local Mesh Modification

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



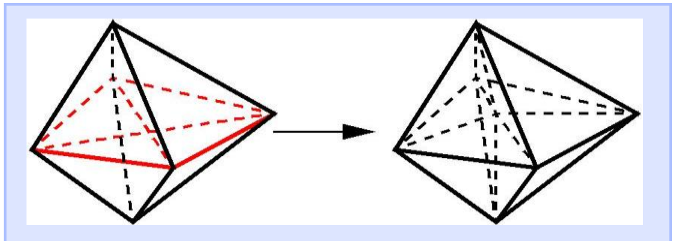
Edge split

face split



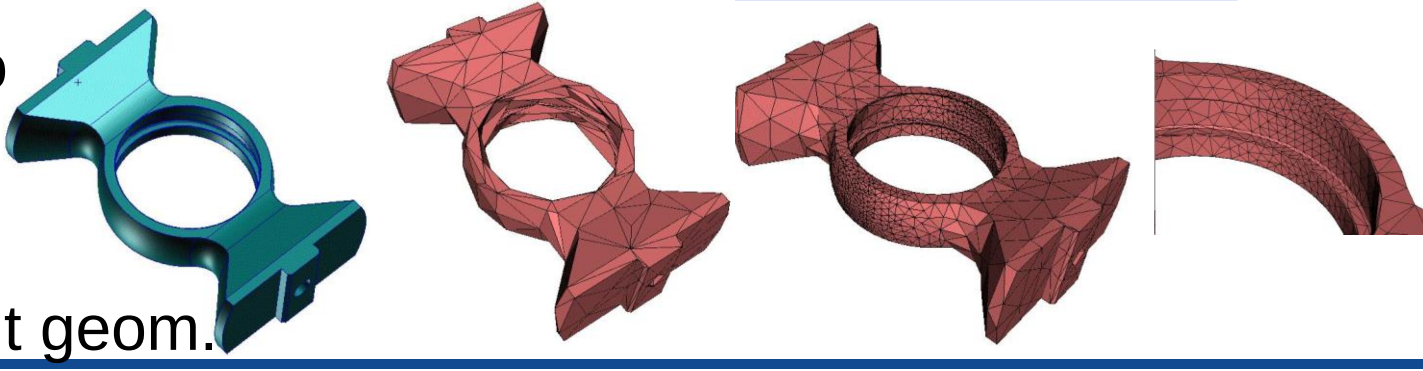
Edge collapse

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



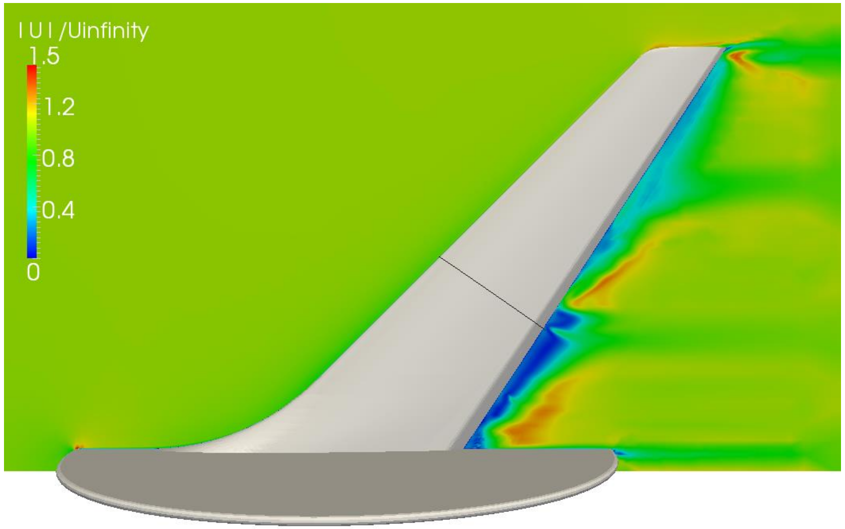
Double split collapse to remove sliver

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



Attached Parallel Fields (APF)

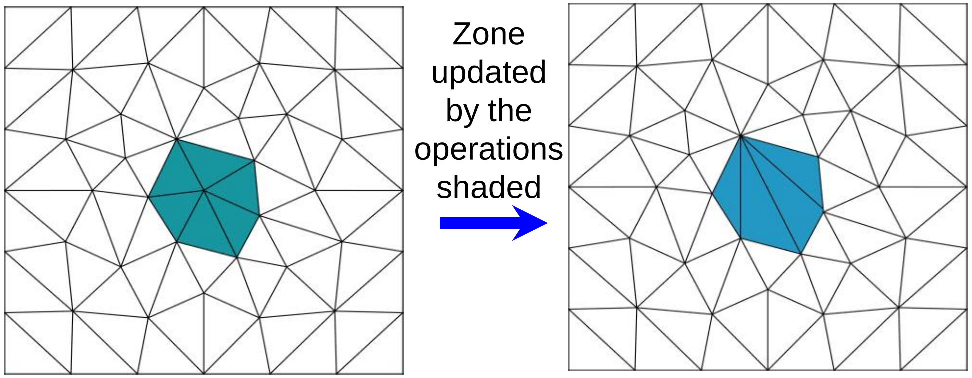
- Attached Parallel Fields (APF) development underway
- Effective storage of solution fields on meshes
- Supports operations on the fields
  - Interrogation
  - Differentiation
  - Integration
  - Interpolation/projection
- Recent efforts
  - Adaptive expansion of Fields from 2D to 3D in M3D-C1
  - History-dependent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



Local Solution Transfer

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



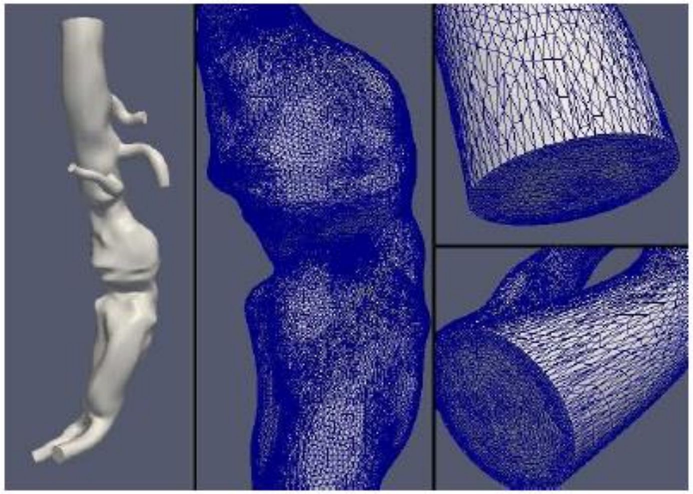
Zone updated by the operations shaded

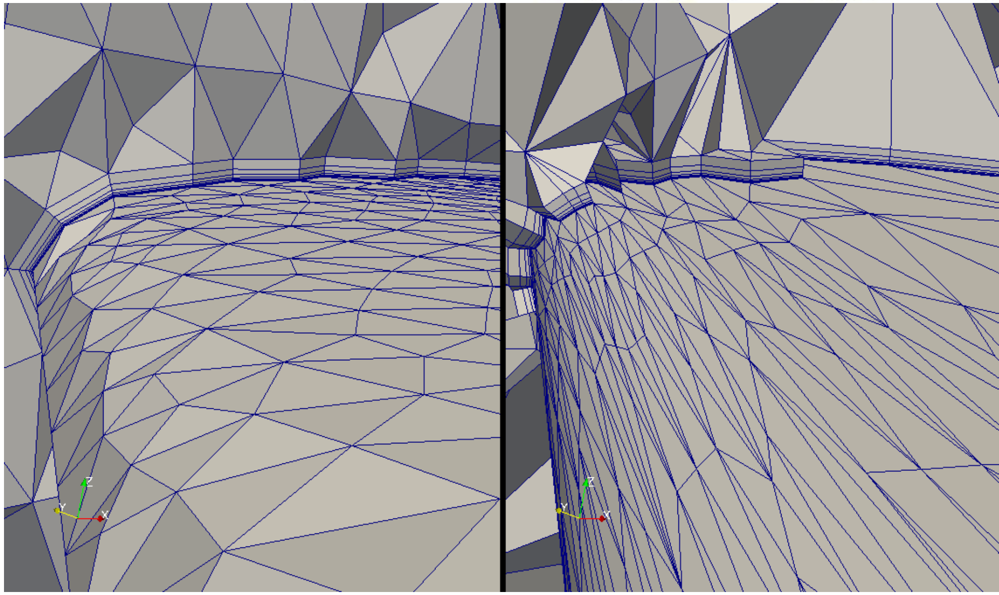
Before collapse

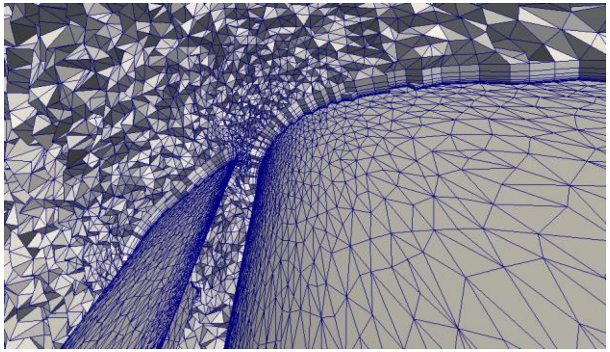
after collapse

Mesh Adaptation Status

- Applied to very large scale models – up to 92 billion elements on ¾ million cores
- Local solution transfer supported through callback
- Effective storage of solution fields on meshes
- Supports adaptation with boundary layer meshes

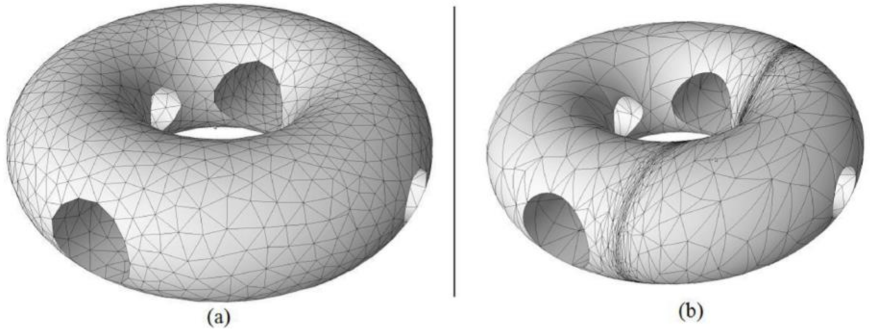


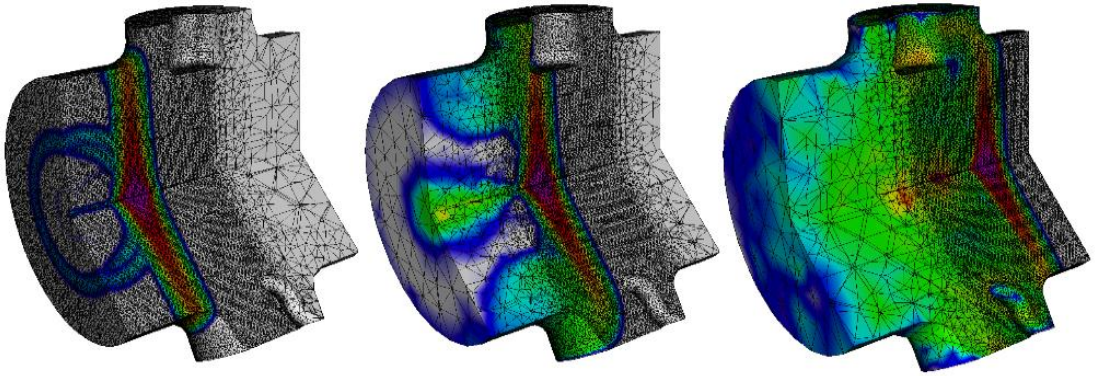


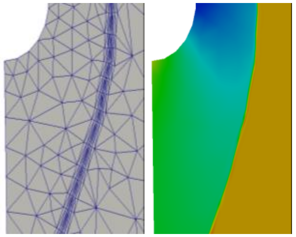


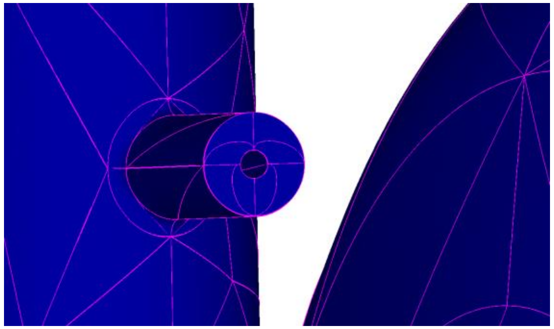
Mesh Adaptation Status

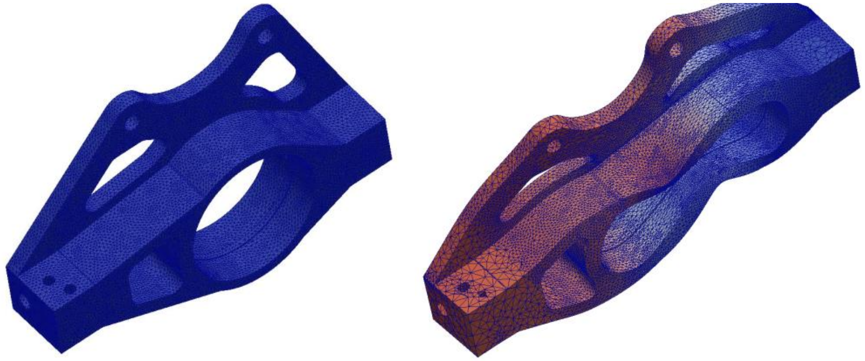
- Supports adaptation of curved elements
- Adaptation based on multiple criteria, examples
  - Level sets at interfaces
  - Tracking particles
  - Combination of mesh errors and element shape evolving geometry











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

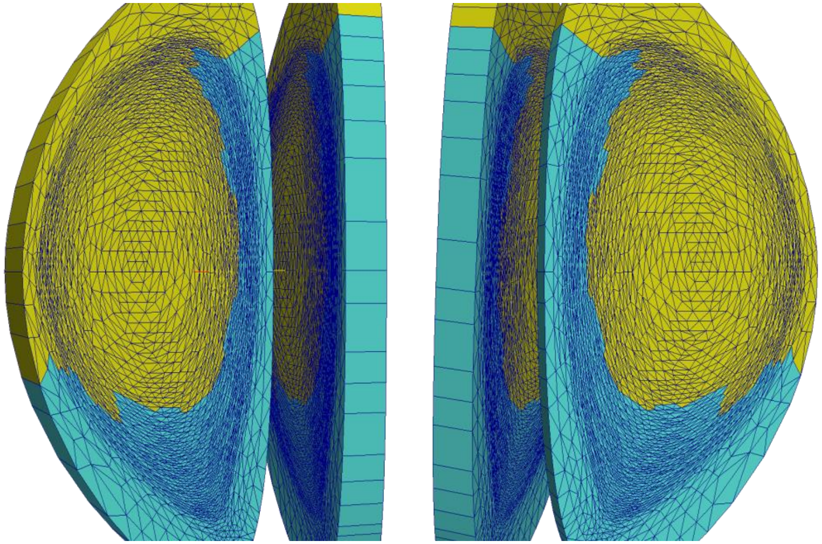
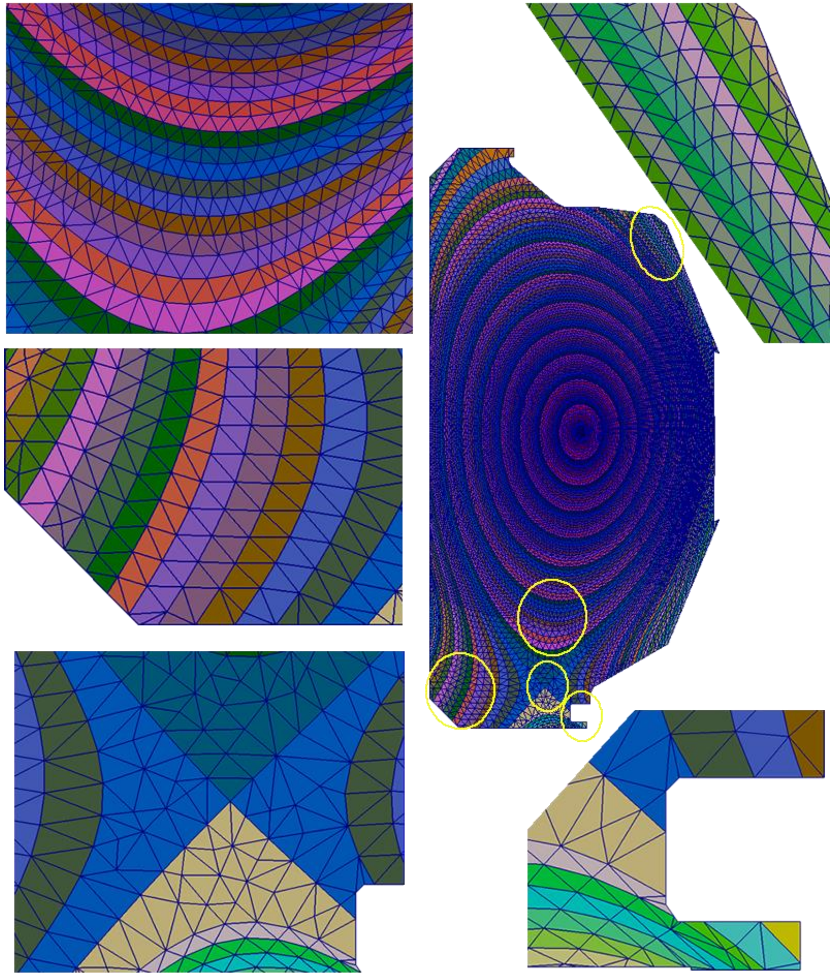


Fig: 3D mesh constructed from 64 2D planes on 12288 processes [1] (only the mesh between selected planes shown)

[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

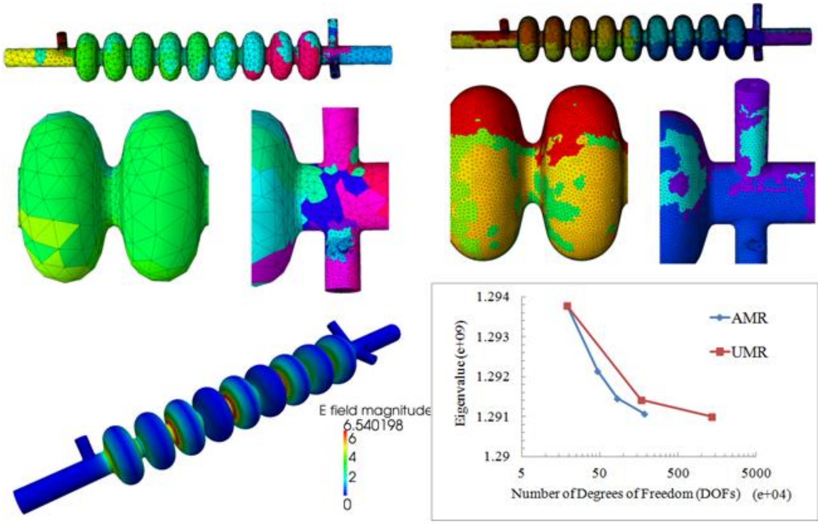
Highlight: Unstructured Mesh Techniques for Edge Plasma Fusion Simulations

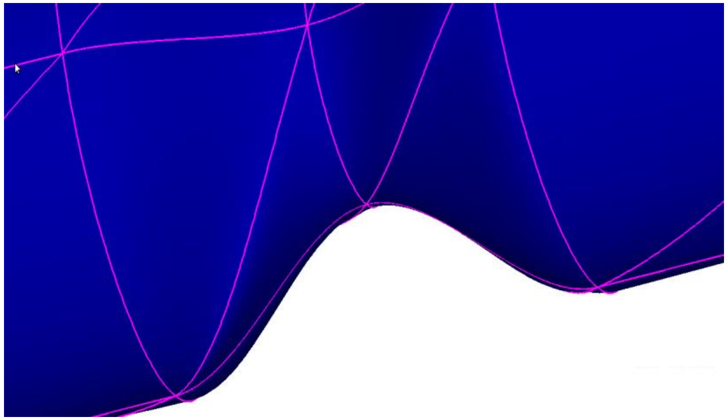
- EPSI PIC coupled to mesh simulation requires high quality meshes meeting a strict set of layout constraints
  - Existing method took >11 hours and mesh did not have desired quality
  - FASTMath meshing technologies put together to produce better quality meshes that meet constraints
  - Run time reduced by a factor of >60 to under 10 minutes for finest mesh
- 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

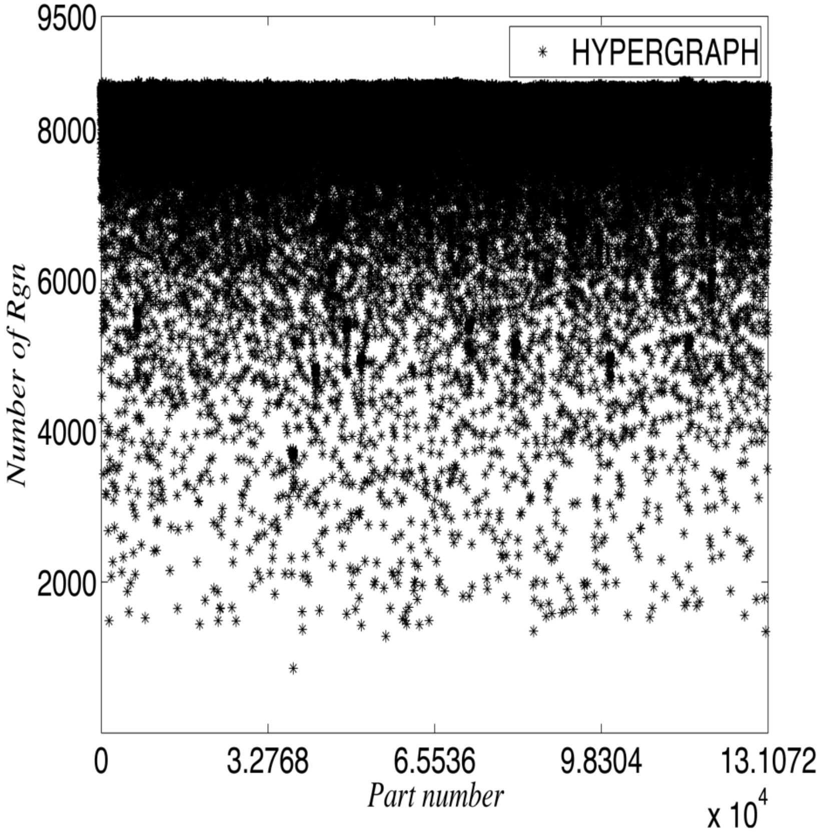
- Provide parallel mesh modification procedure capable of creating/adapting curved mesh geometry
- Parallel mesh adaptation procedure developed that supports quadratic curved meshes
- Ongoing efforts to support higher order G1 mesh geometry
- The procedure integrated with high-order electro-magnetic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



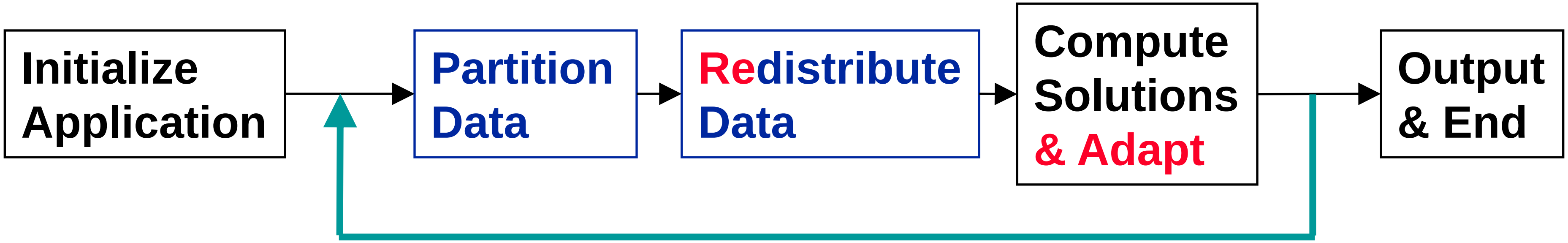


Dynamic Load Balancing

- Purpose: to rebalance load-imbalanced mesh during mesh modification
  - 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



Dynamic Load Balancing



Initialize Application

Partition Data

Redistribute Data

Compute Solutions & Adapt

Output & End

- 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 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.

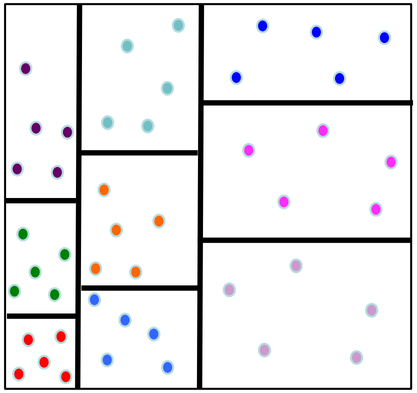
Dynamic:

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

Zoltan/Zoltan2 Toolkits: Partitioners

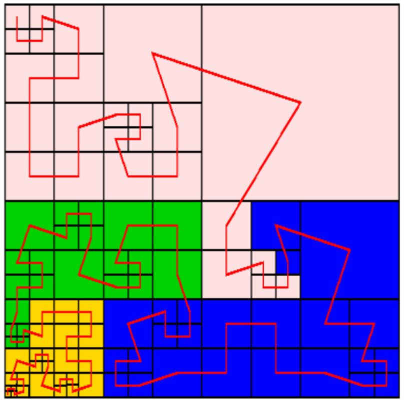
Suite of partitioners supports a wide range of applications; no single partitioner is best for all applications.

Geometric

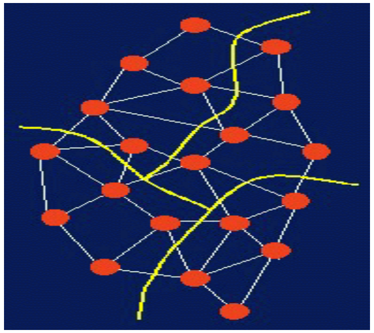


Recursive Coordinate Bisection
Recursive Inertial Bisection
Multi-Jagged Multi-section

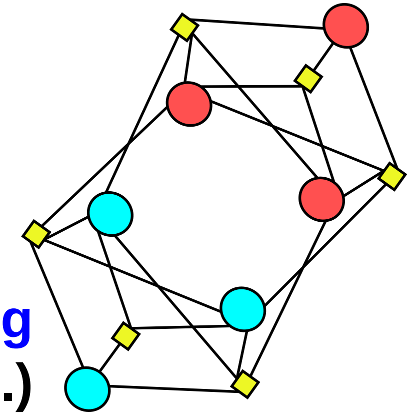
Space Filling Curves



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

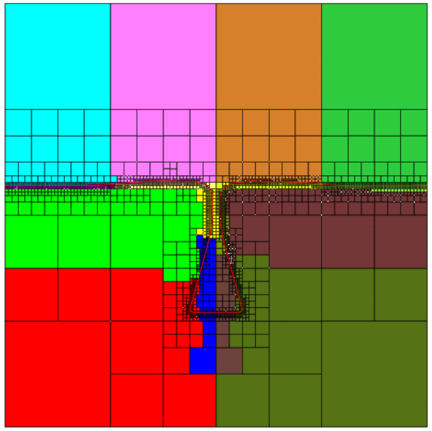
- RCB/RIB: Compute cutting planes that recursively bisect workloads
- MJ: Multi-section instead of bisection to reduce cost of partitioning
- SFC: Partition linear ordering given by space-filling curve

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
- SFC: Linear ordering may improve cache performance

Disadvantages:

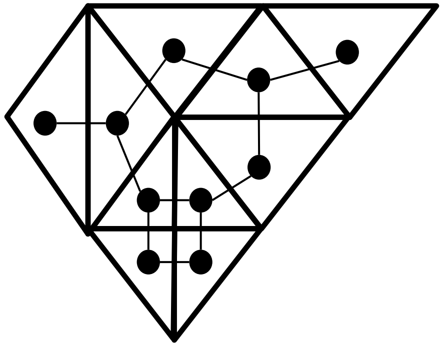
- No explicit control of communication costs
- Geometric coordinates needed

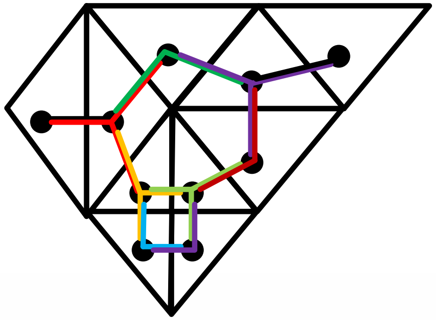


Topology-based Partitioners

Goal: Balance work while minimizing data dependencies between parts

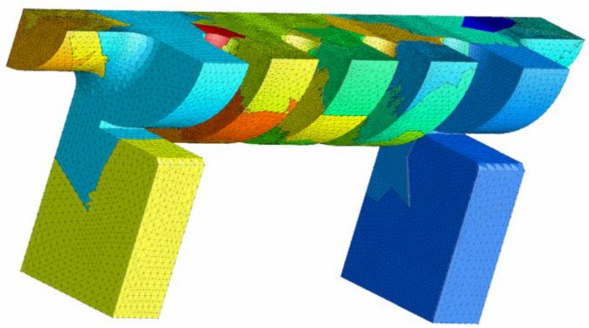
- Represent data with vertices of graph/hypergraph
- Represent dependencies with graph/hypergraph edges





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

Partitioning using Mesh Adjacencies (ParMA)

Mesh and partition model adjacencies directly used

- All mesh entities can be considered
- Any adjacency can be obtained in O(1) time
- Directly account for multiple entity types – important for the solve process – most computationally expensive step
- Avoid graph construction
- Easy to use with diffusive procedures
- Applications to Date
  - Partition improvement to account for multiple entity types and cost functions – improved scalability of solvers
  - Use for improving partitions on really big meshes

ParMA – Multi-Criteria Partition Improvement

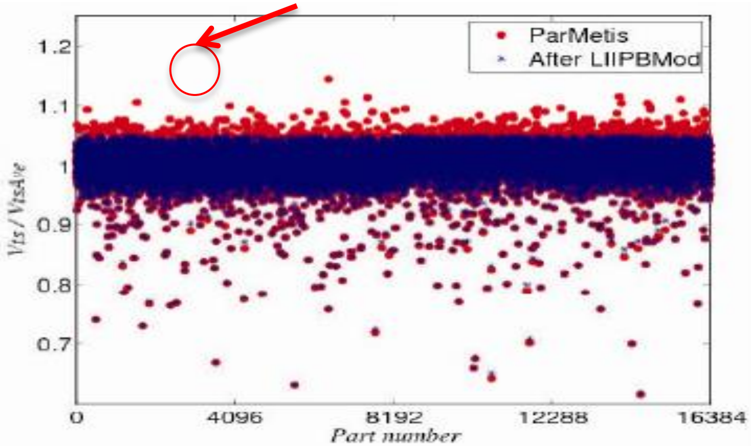
- Improved scalability of the solve by accounting for balance of multiple entity types – eliminate spikes
- Input:
  - Priority list of entity types to balance (region, face, edge, vertex)
  - Mesh with entity communication, computation and migration weights
- Algorithm:
  - From high to low priority if separated by '>' and From low to high dimension entity types if separated by '='
    - Compute migration schedule (Collective), Select regions for migration (Embarrassingly Parallel), Migrate selected regions (Collective)
  - Ex) "Rgn>Face=Edge>Vtx" is the user's input
    - Step 1: improve balance for mesh regions
    - Step 2.1: improve balance for mesh edges
    - Step 2.2: improve balance for mesh faces
    - Step 3: improve balance for mesh vertices

ParMA Application Partition Improvement

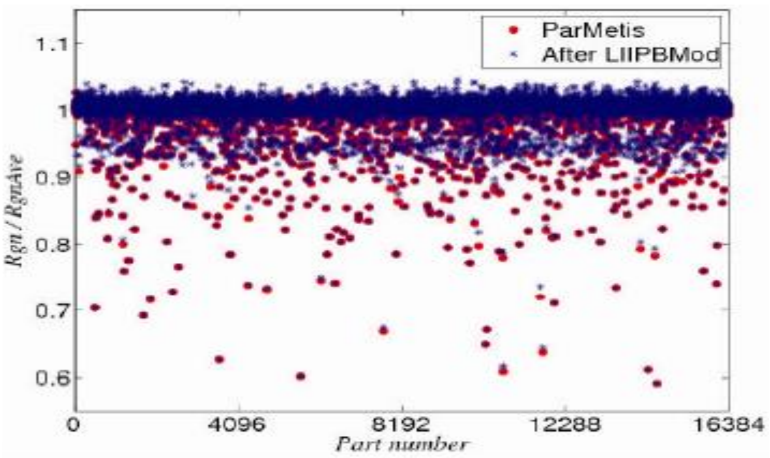
Example of C0, linear shape function finite elements

- 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
- Diffusive application of ParMA knocks spikes down – common to see 10% increase in strong scaling

***dof* imbalance reduced - 14.7% to 4.92%**

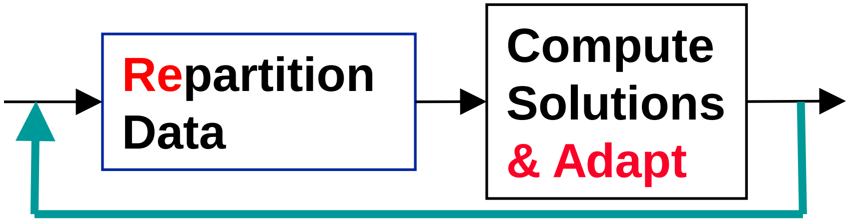


***element* imbalance increased - 2.64% to 4.54%**



Predictive Load Balancing

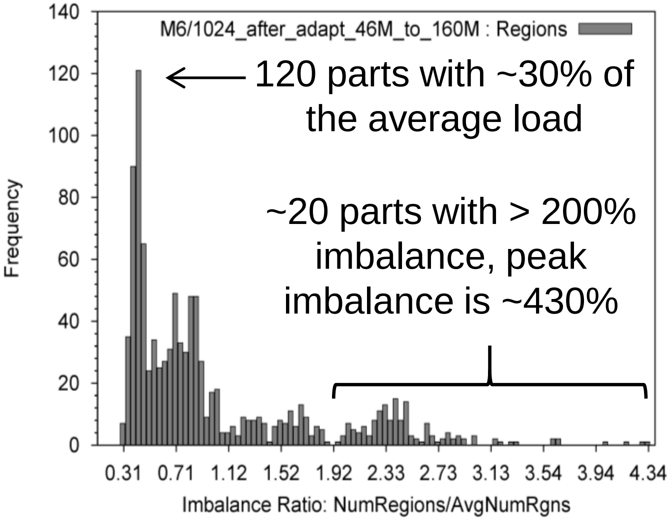
- Mesh modification before load balancing can lead to memory problems - common to see 400% increase on some parts



Repartition Data

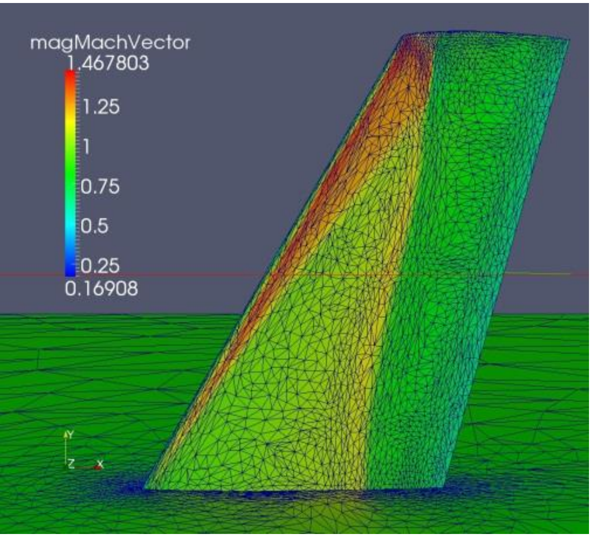
Compute Solutions & Adapt

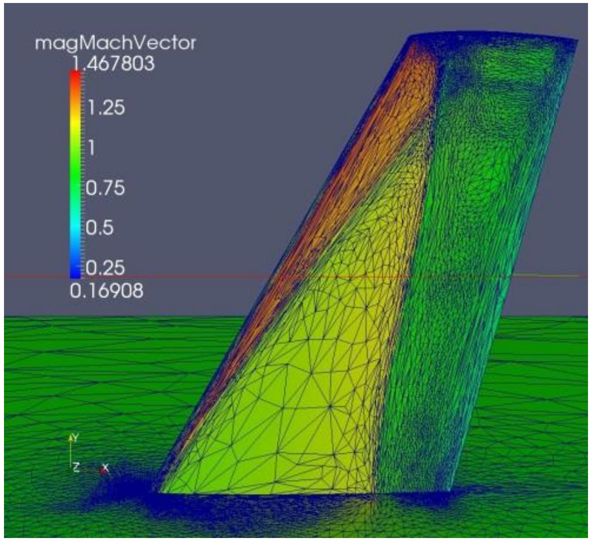
- Employ predictive load balancing to avoid the problem
  - Assign weights based on what will be refined/coarsened
  - Apply dynamic load balancing using those weights
  - Perform mesh modifications



120 parts with ~30% of the average load

~20 parts with > 200% imbalance, peak imbalance is ~430%





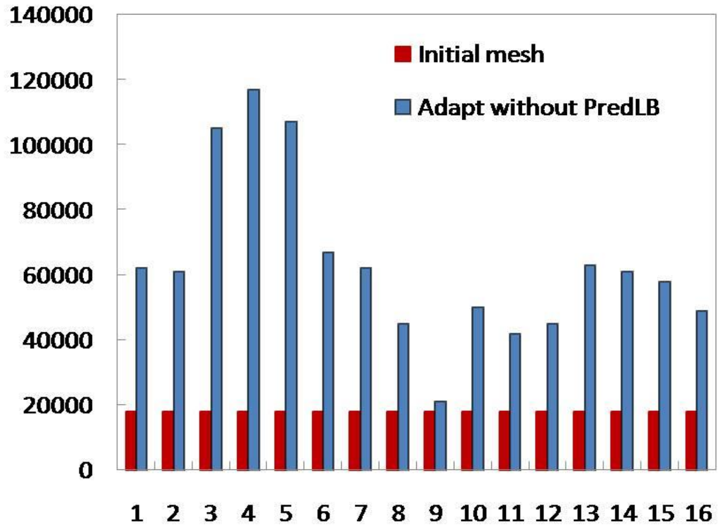
Histogram of element imbalance in 1024 part adapted mesh on Onera M6 wing if no balancing applied prior to adaptation.

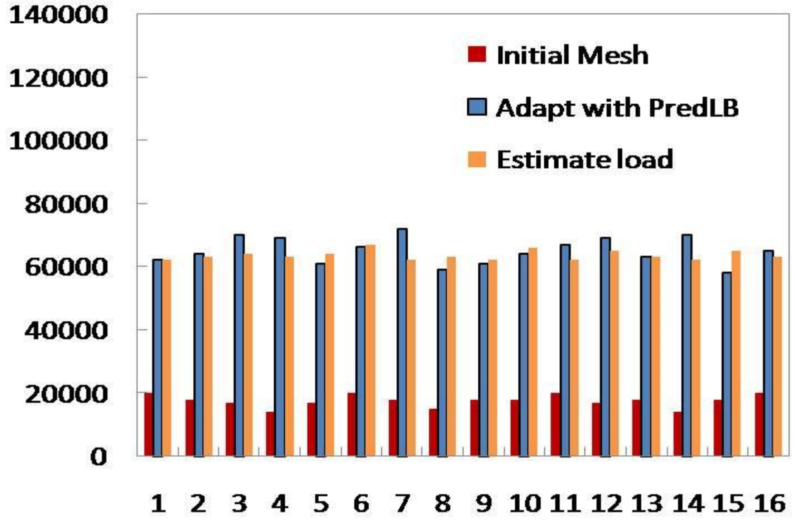
Predictive Load Balancing - Algorithm

- Mesh metric field decomposed into orthogonal directions (e_1,e_2,e_3) and desired length (h_1,h_2,h_3) in each direction.
- The volume of desired element (tetrahedron): $h_1h_2h_3/6$
- Estimate number of elements to be generated:

$$num = \frac{R\_volume}{\sum_{p=1}^{n_{en}} h_1(p)h_2(p)h_3(p)/6n_{en}}$$

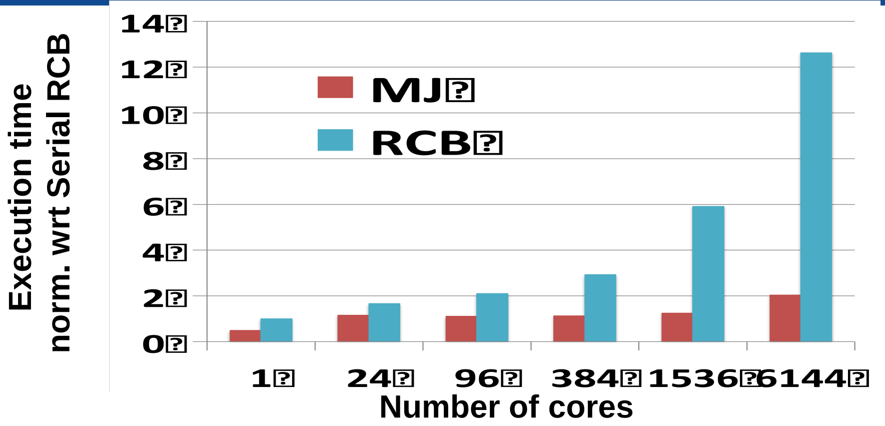
- Load balance based on weighted graph nodes



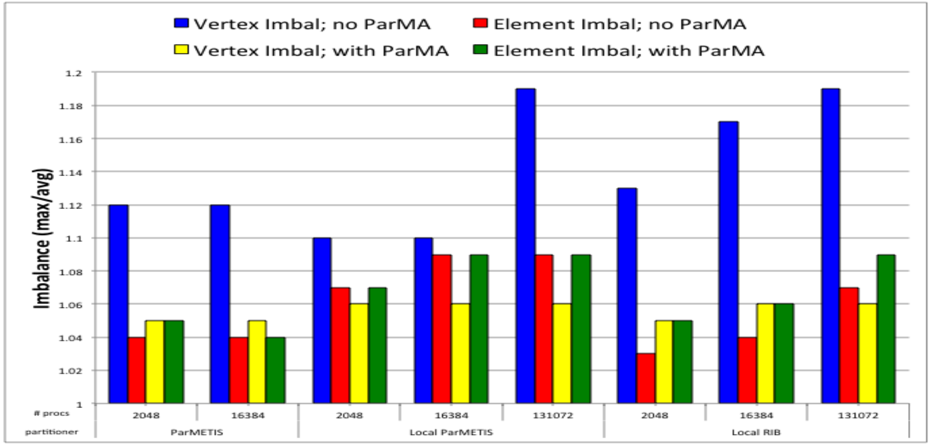


Dynamic Load Balancing to Ensure the Ability of Applications to Achieve and Maintain Scalability

- Results/Impact
  - Zoltan’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



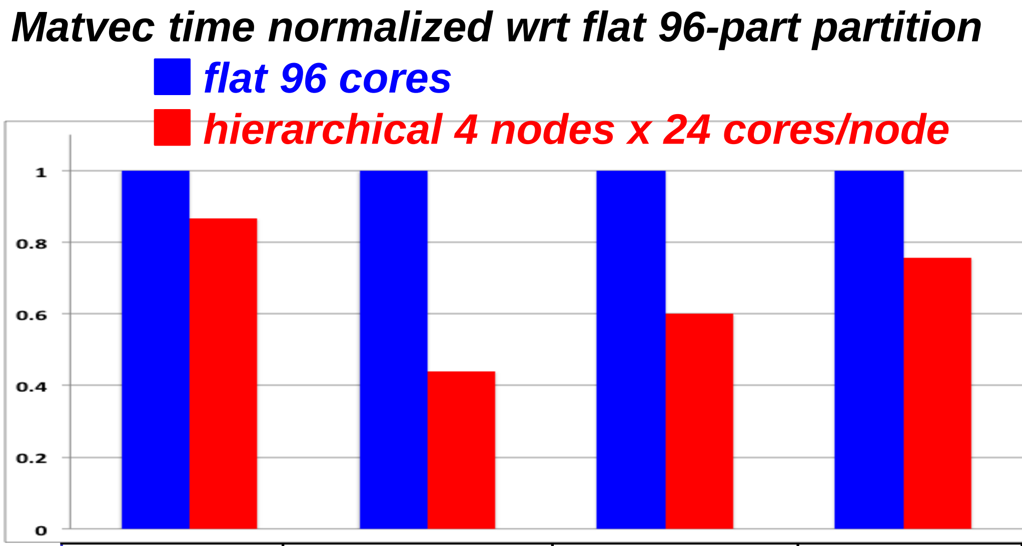
Reduced data movement in MultiJagged partitioner enables better scaling than Recursive Coordinate Bisection on NERSC’s Hopper.



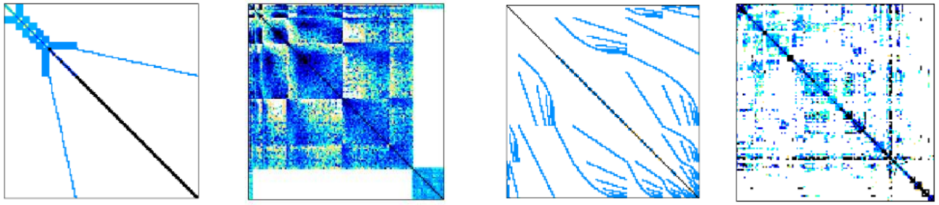
For very little cost, ParMA improves application scalability by dramatically decreasing vertex imbalance while maintaining element balance.

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



Architecture-Aware Geometric Task Placement in Zoltan2

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

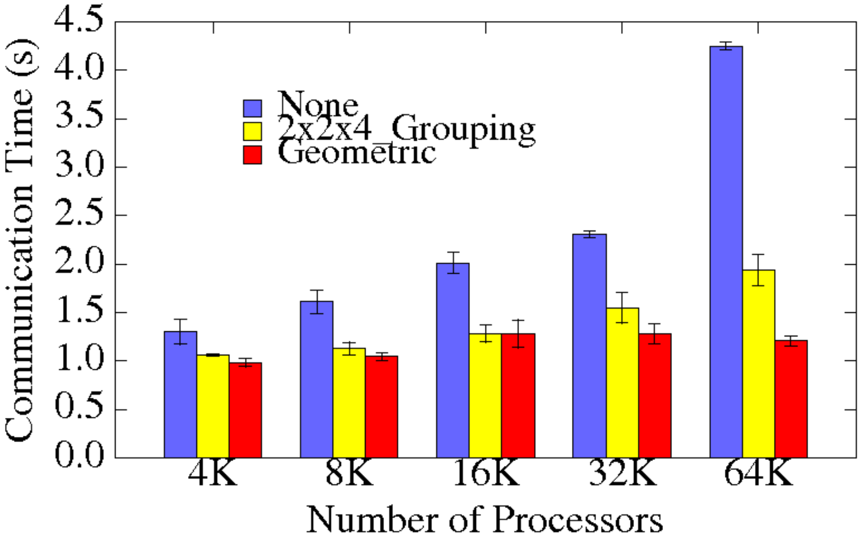
- Especially important in non-contiguous node allocations (e.g., Hopper Cray XE6)

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 custom 2x2x4 task-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

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 problems in multi-dimensions.

MOAB Discretization Manager

- Provides ability to discretize physics PDE with FEM/FD based on a native MOAB mesh leveraging scalable PETSc solvers.
- Design resembles structured (DMDA) and unstructured (DMPlex) interfaces; software productivity.
- Support both strided and interleaved access of field components; Opens up better preconditioning strategies.
- Analyze efficient unstructured mesh traversal, FD/FEM-type operator assembly for relevant problems in multi-dimensions.
- Optimized computation of physics residuals using PETSc Vec that reuses the contiguous memory provided by MOAB tags.
- Capabilities to define field components, manage degrees-of-freedom, local-to-global transformations.

- The implementation is part of the latest PETSc 3.5 release (DM)

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

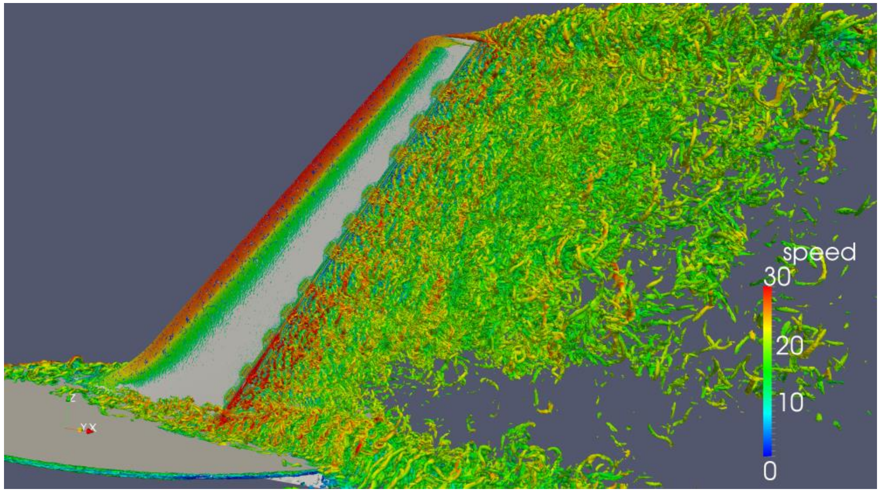
MOAB Discretization Manager – Examples

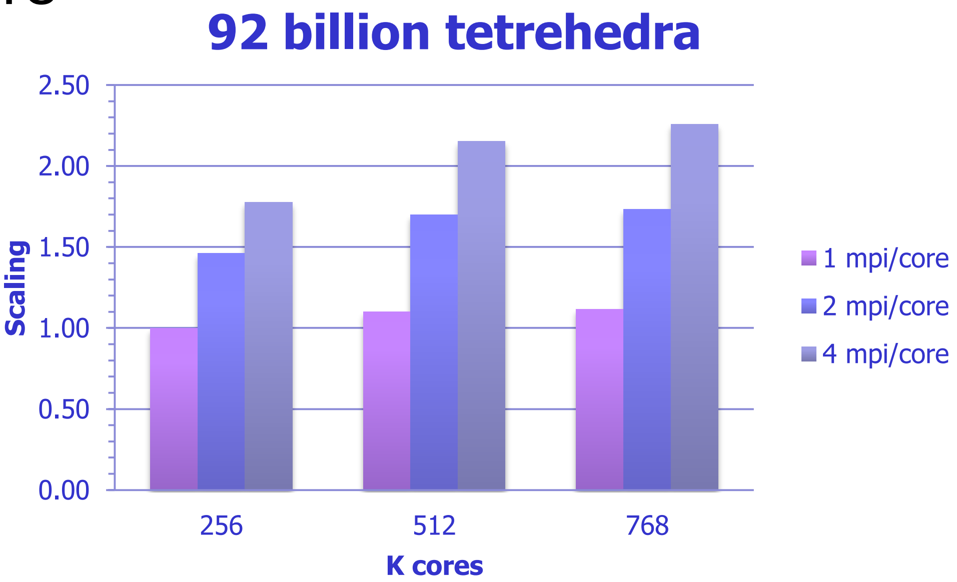
- Motivator: PSI (Plasma Surface Interactions) project for integration with XOLOTL code.
  - Utilize underlying array-based mesh data structures to perform high-order multi-component solver based on PETSc.
  - Reduce total memory use by sharing vector spaces and allowing block filling of coupled component terms in the linear operator.
  - Fast stiff ODE-solvers for reaction-diffusion equations via IMEX methods (PETSc) that can accommodate sparse coupling between the components.
- Some relevant tutorial examples in PETSc:
  - Multi-component 1-d time-dependent Brusselator reaction-diffusion PDE FEM solver in 1-d. (ts/examples/tutorials/ex35.c)
  - A 2-D, verifiable Diffusion-Reaction FEM steady state solver. (ksp/ksp/examples/tutorials/ex35.c)

Massively Parallel Unstructured Mesh Solver (PHASTA)

Implicit, Adaptive Grid CFD

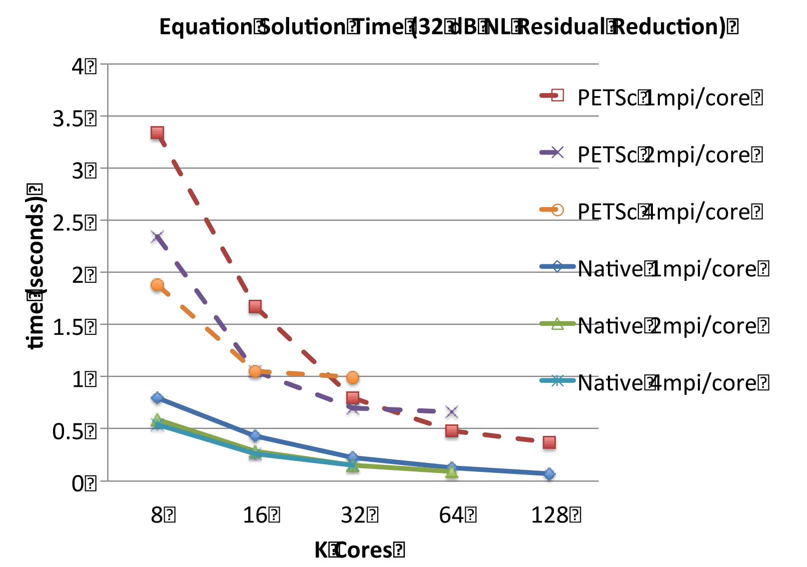
- Extreme Scale Applications:
  - Aerodynamics flow control
  - Multiphase flow
- Full Machine Strong scaling
  - Variable MPI processes/core
  - 92 Billion tetrahedra
  - 262144 to 3,145,728 parts
  - 1/core 100% scaling
  - 2/core 146-155% scaling
  - 4/core 178-226% scaling

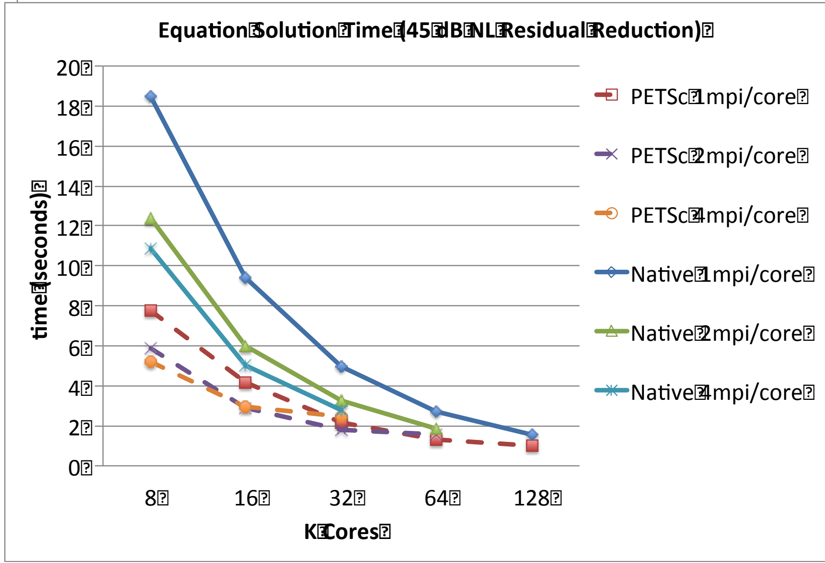




PHASTA/PETSc Coupling

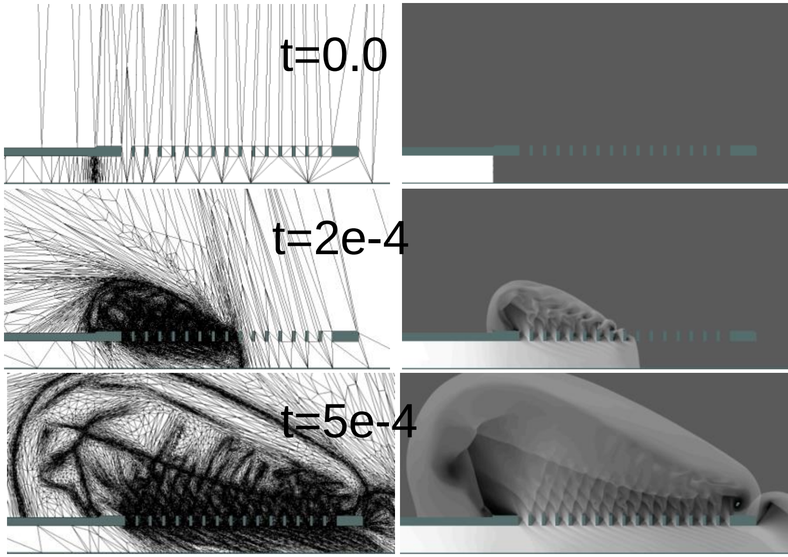
- PETSc functions assemble LHS and RHS
- PETSc MatAssembly performs additional step of building globally complete matrix
- Relative efficiency depends on solve tolerance – tighter tolerances more efficient with PETSc
- Multiple processes per core currently benefit native solver more than PETSc
- MatAssembly times identified as a scaling bottleneck to be improved

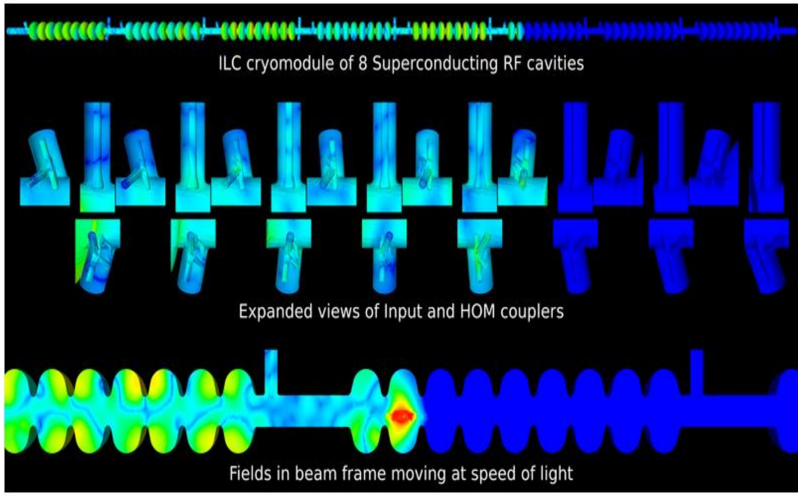




Creation of Parallel Adaptive Loops

- 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

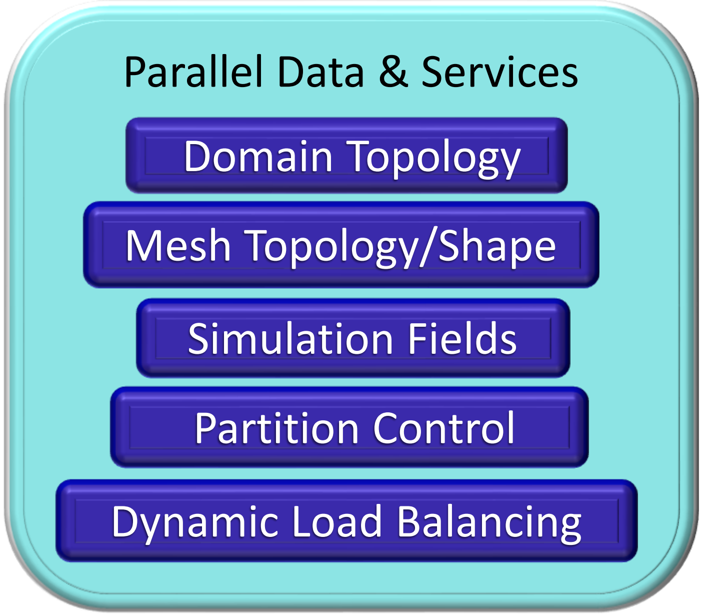



ILC cryomodule of 8 Superconducting RF cavities
Expanded views of Input and HOM couplers
Fields in beam frame moving at speed of light

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



Parallel Data & Services

Domain Topology

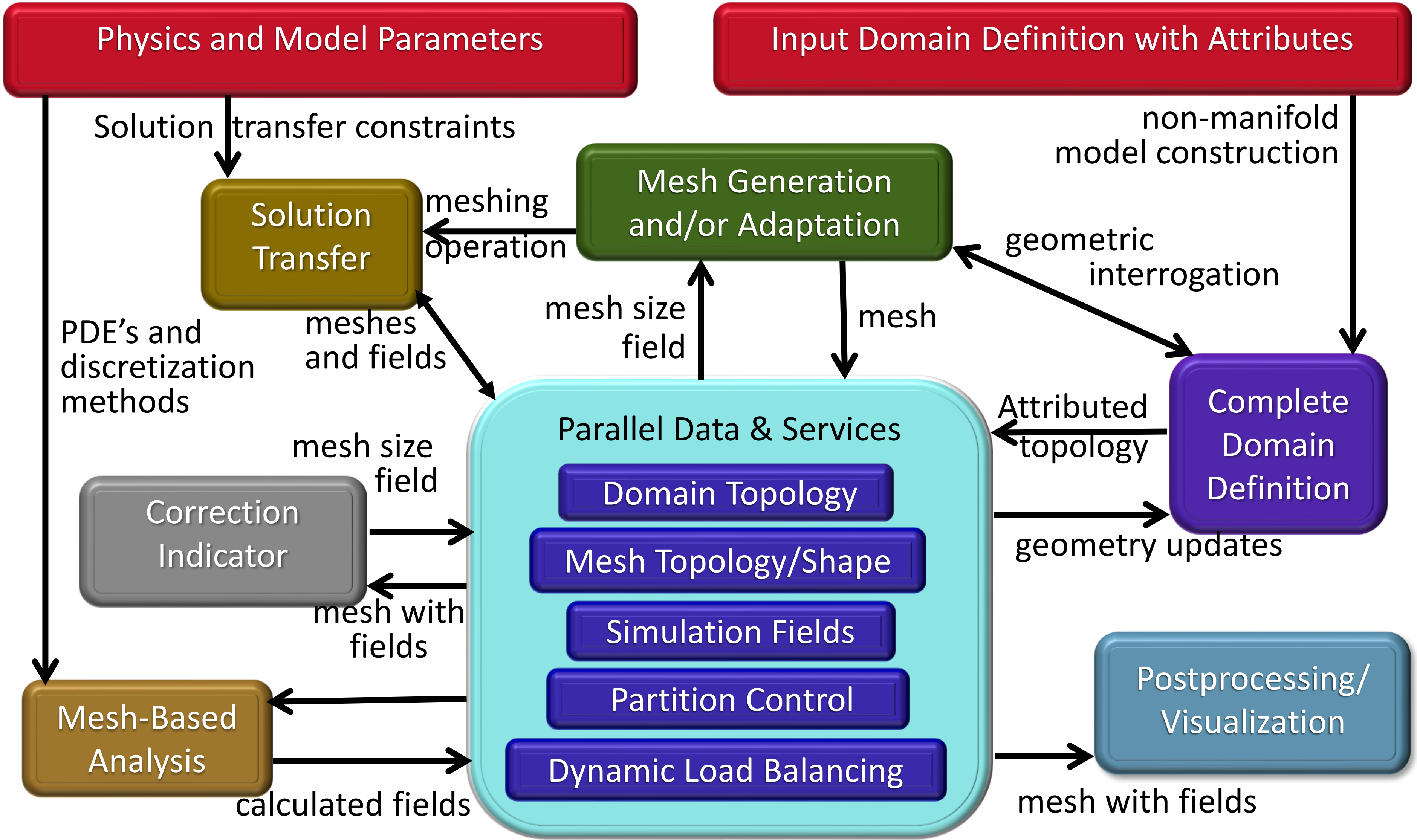
Mesh Topology/Shape

Simulation Fields

Partition Control

Dynamic Load Balancing

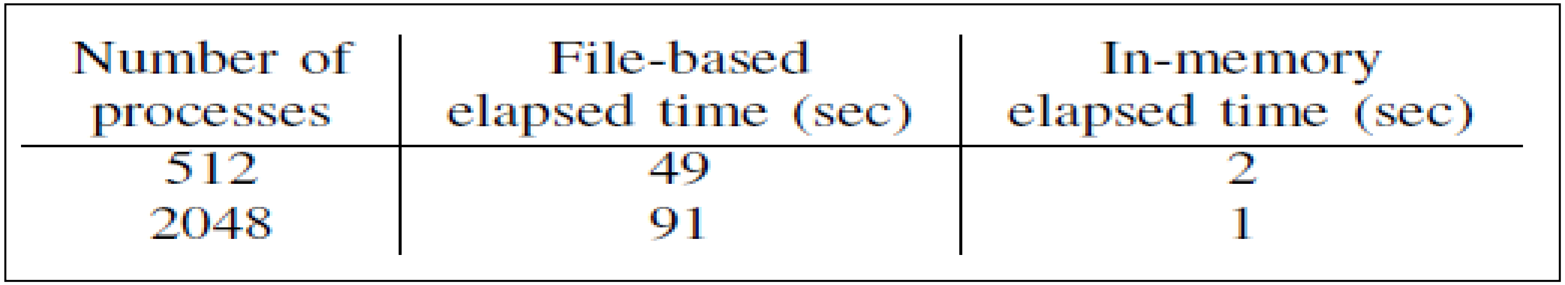
Components in Parallel Adaptive Analysis



In-Memory Coupling of Simulation Components

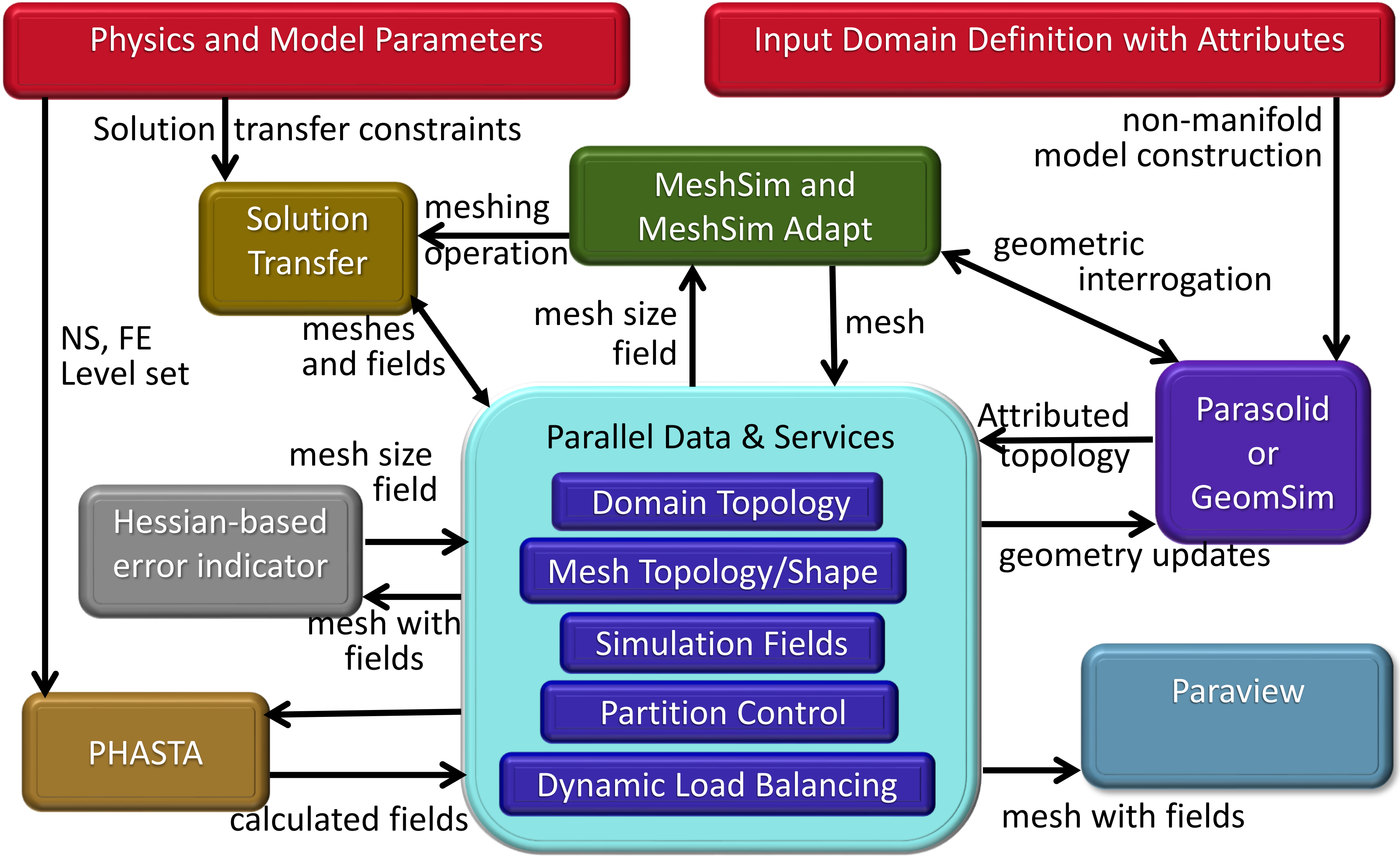
File transfer a serious bottleneck in parallel simulation workflows

- All core parallel data and services accessed through APIs
- In-memory integration approach uses APIs
  - Migration from file-based components to in-memory
    - Modify/extend components by wrapping data structures with APIs for:
      - read/write
      - memory management
      - inter-language coupling; typically FORTRAN and C
- 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

Adaptive Active Flow Control



Physics and Model Parameters

Input Domain Definition with Attributes

Solution transfer constraints

non-manifold model construction

MeshSim and MeshSim Adapt

Solution Transfer

meshing operation

geometric interrogation

NS, FE Level set

meshes and fields

mesh size field

mesh

Parasolid or GeomSim

Attributed topology

Parallel Data & Services

Domain Topology

Mesh Topology/Shape

Simulation Fields

Partition Control

Dynamic Load Balancing

mesh size field

Hessian-based error indicator

mesh with fields

geometry updates

PHASTA

calculated fields

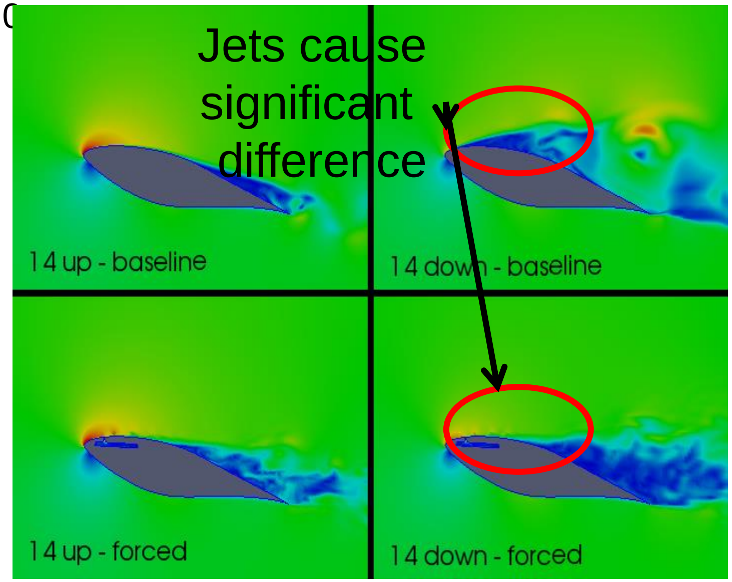
Paraview

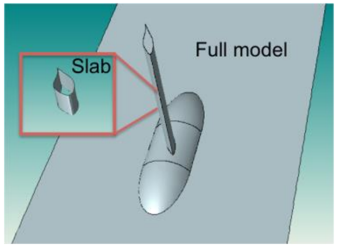
mesh with fields

Adaptive Active Flow Control

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

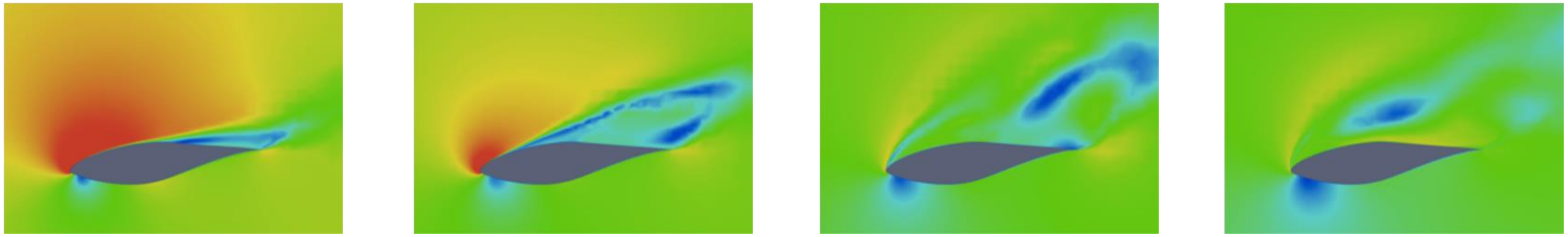
- Slab model – pitch rate of 10Hz
- Baseline (without jets) and forced/controlled (with jets)



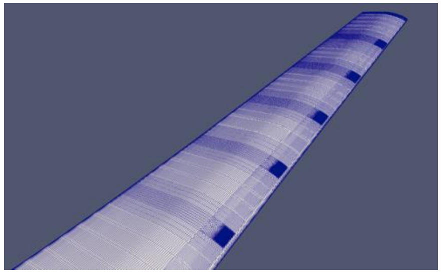


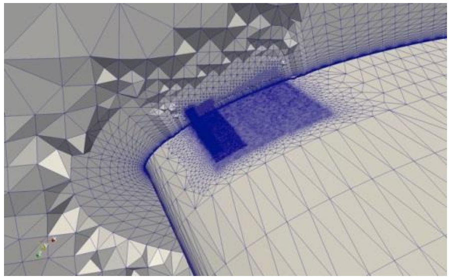
Case\AoA (L/D)	14 Up	14 Down
Baseline	10.2236	3.2286
Forced	10.6265	**9.9921**

15 m/s

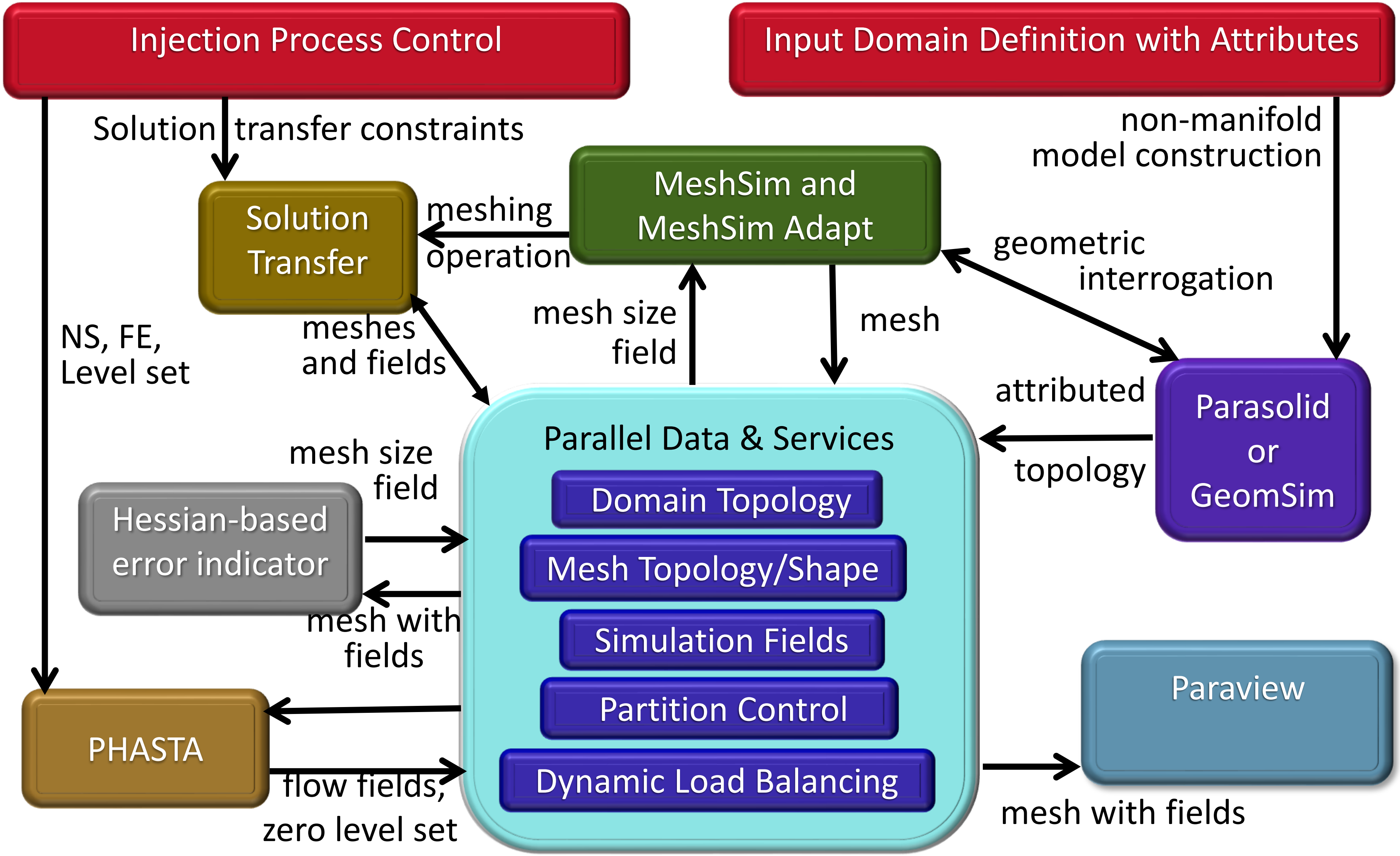


Leading-edge synthetic jets: 5 along span



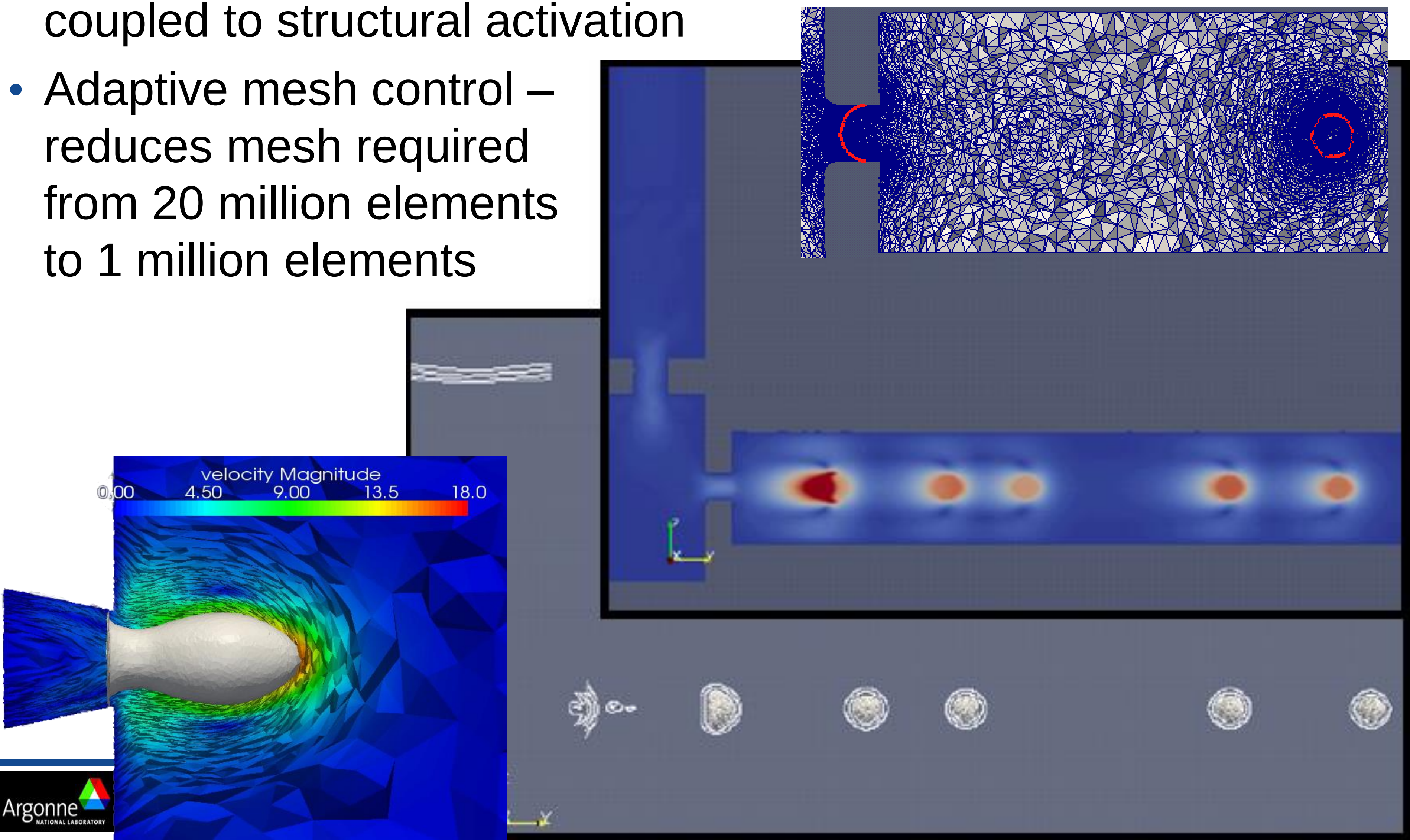


Adaptive Two-Phases Flow

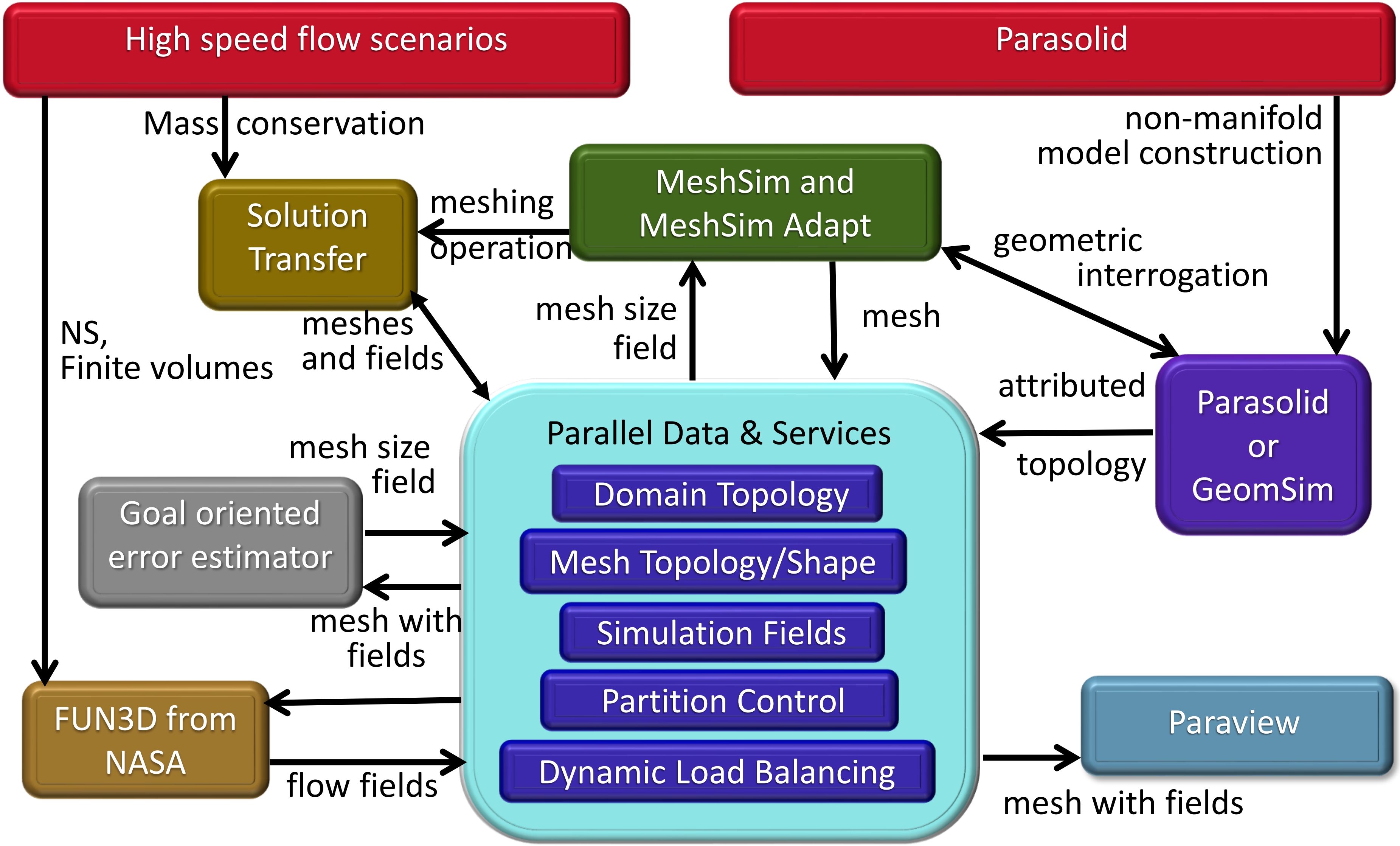


Adaptive Two-Phases Flow

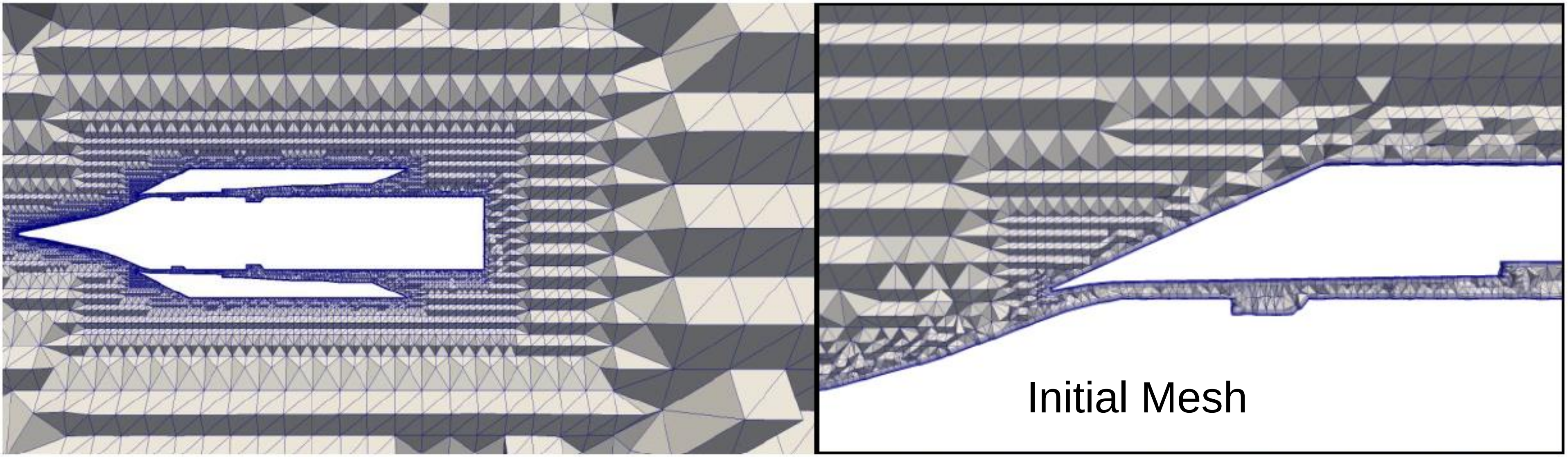
- Two-phase modeling using level-sets coupled to structural activation
- Adaptive mesh control – reduces mesh required from 20 million elements to 1 million elements



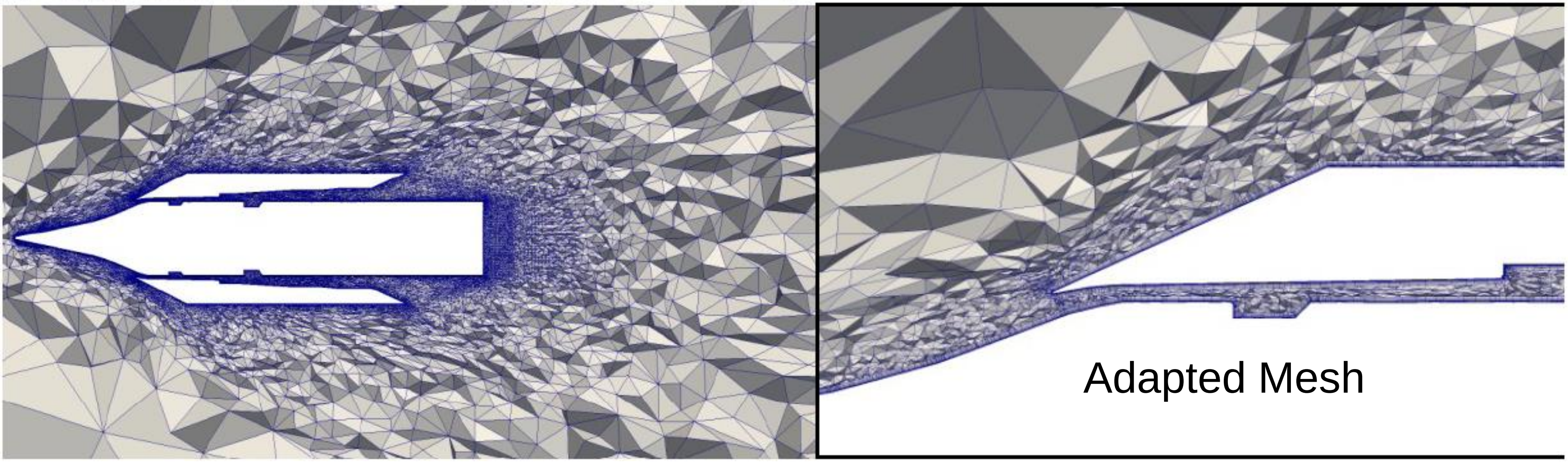
Aerodynamics Simulations



Application Result - Scramjet Engine

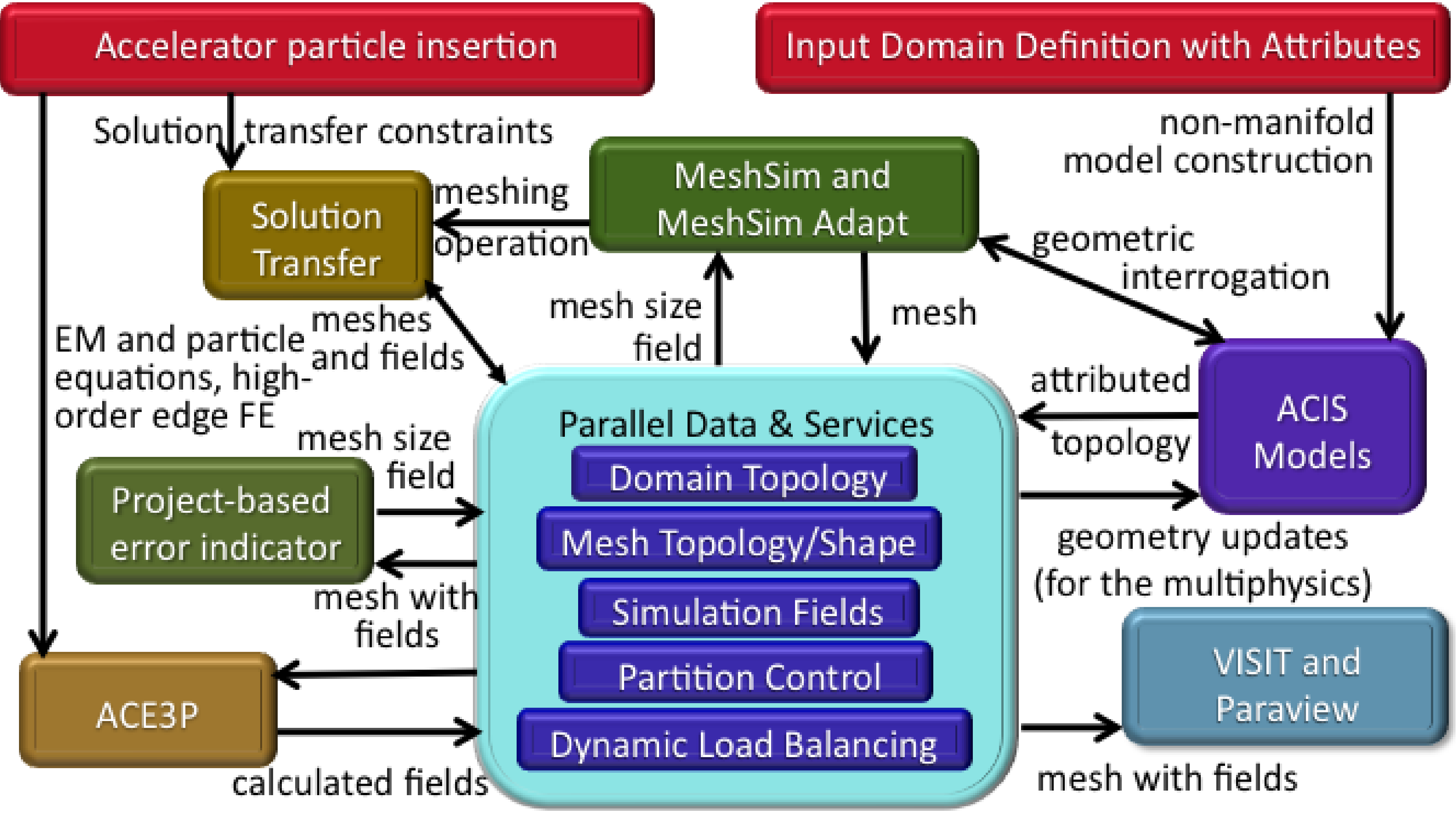


Initial Mesh



Adapted Mesh

Electromagnetics Analysis



Accelerator particle insertion

Input Domain Definition with Attributes

Solution transfer constraints

non-manifold model construction

Solution Transfer

meshing operation

MeshSim and MeshSim Adapt

geometric interrogation

mesh size field

mesh

EM and particle equations, high-order edge FE

meshes and fields

attributed topology

ACIS Models

mesh size field

Parallel Data & Services

Domain Topology

Mesh Topology/Shape

Simulation Fields

Partition Control

Dynamic Load Balancing

Project-based error indicator

mesh with fields

geometry updates (for the multiphysics)

VISIT and Paraview

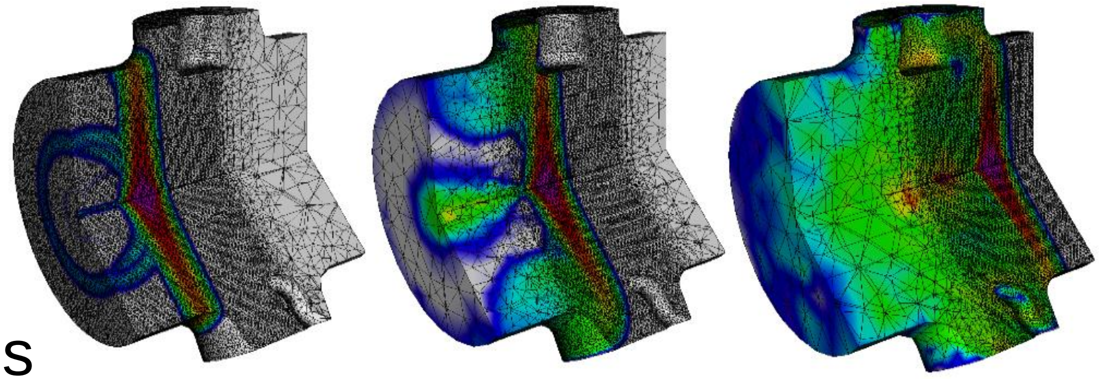
ACE3P

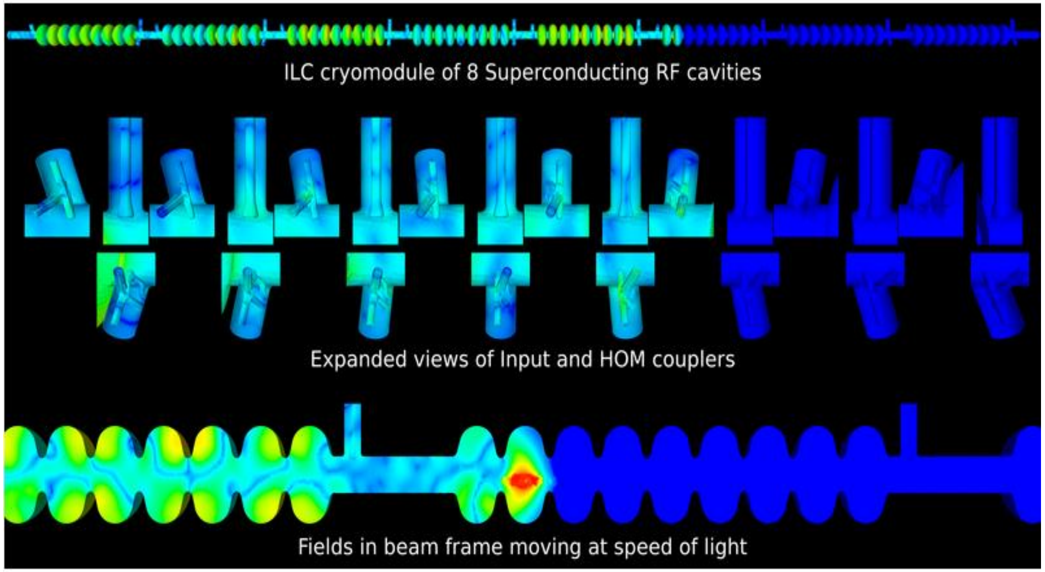
calculated fields

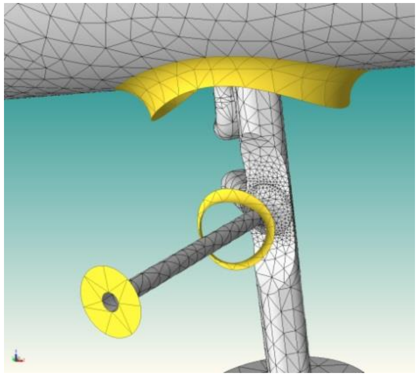
mesh with fields

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

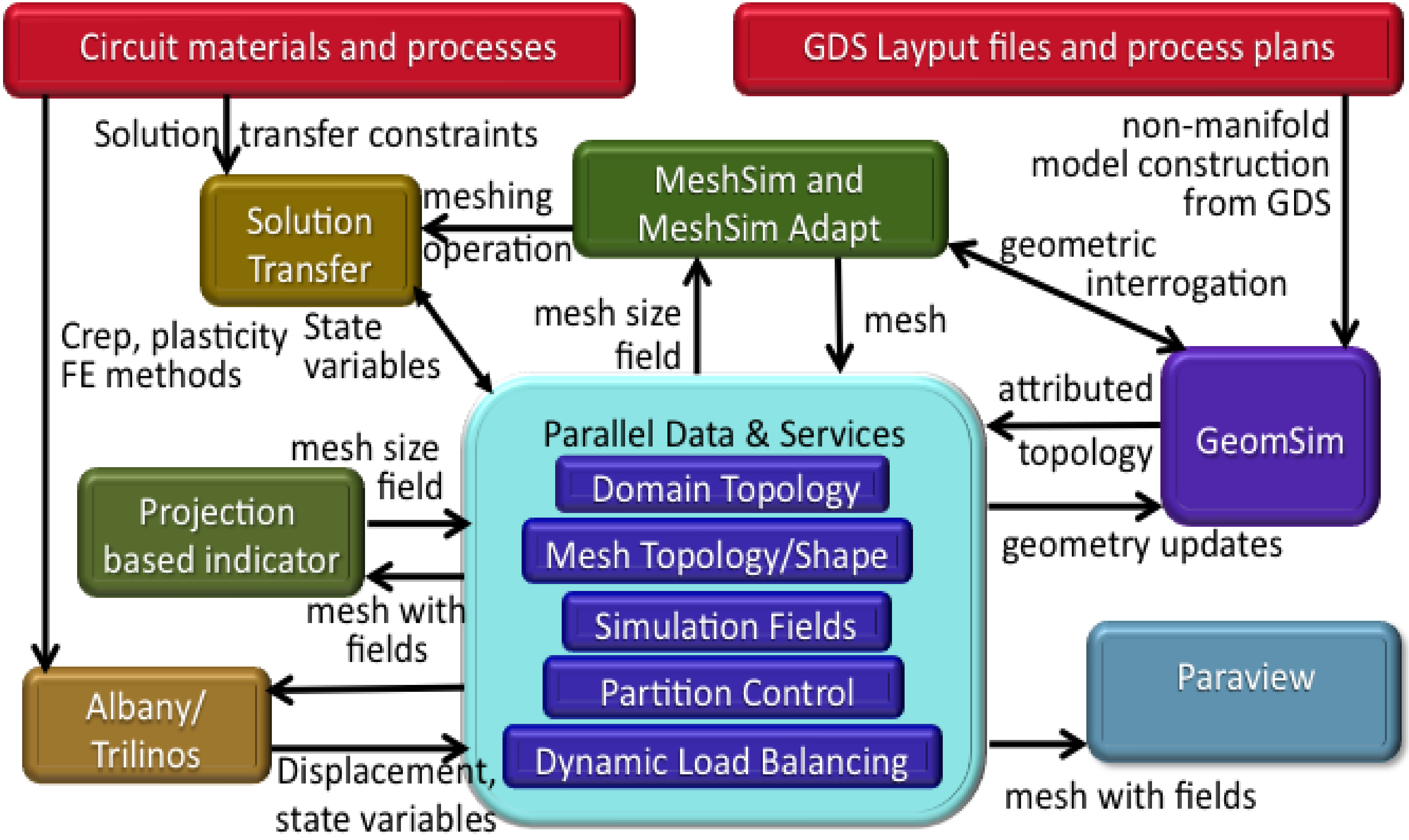



ILC cryomodule of 8 Superconducting RF cavities
Expanded views of Input and HOM couplers
Fields in beam frame moving at speed of light



FASTMATH

Structural Analysis for Integrated Circuits



Circuit materials and processes

GDS Layput files and process plans

Solution transfer constraints

Solution Transfer

meshing operation

MeshSim and MeshSim Adapt

non-manifold model construction from GDS

geometric interrogation

Crep, plasticity FE methods

State variables

mesh size field

mesh

attributed topology

GeomSim

Parallel Data & Services

Domain Topology

Mesh Topology/Shape

Simulation Fields

Partition Control

Dynamic Load Balancing

mesh size field

Projection based indicator

mesh with fields

geometry updates

Albany/ Trilinos

Displacement, state variables

Paraview

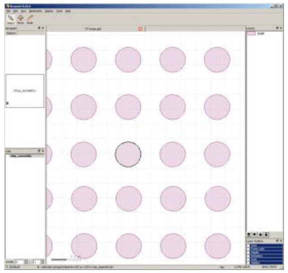
mesh with fields

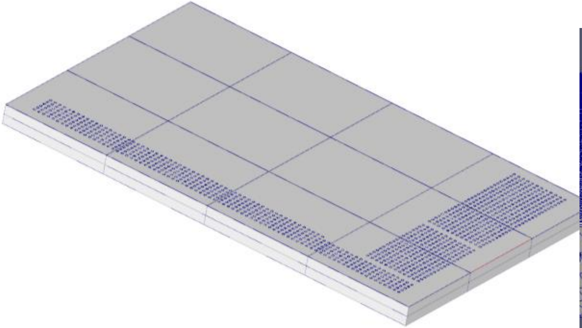
Structural Analysis for Integrated Circuits

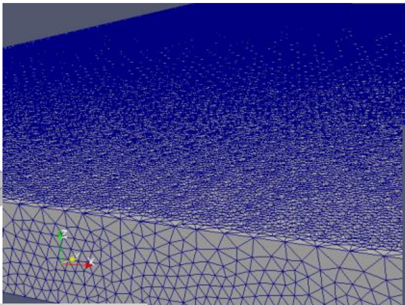
Must construct 3-D non-manifold solid from input geometry

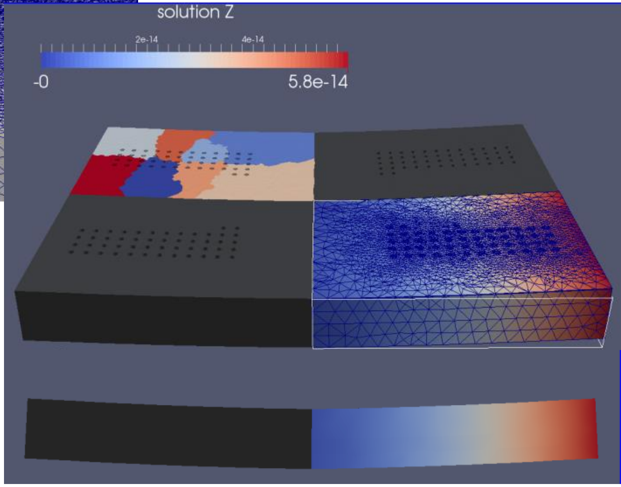
- Input domain defined in terms of 2-D layouts (gdsII/OASIS)
- Third dimension based on process knowledge
- A component has been developed to construct the model

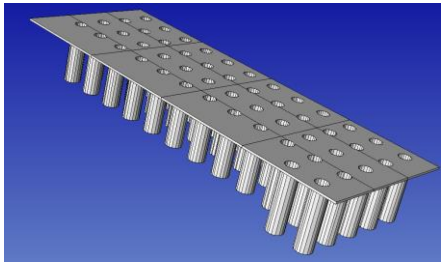
Adaptive loop constructed for thermally loaded case including thin liner



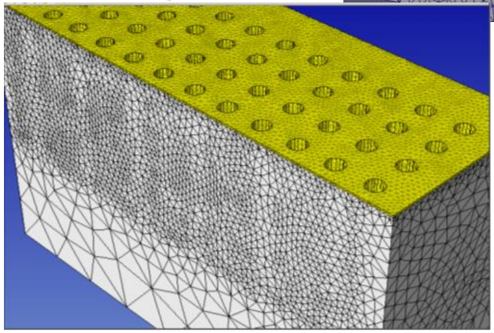








Model of liner film only

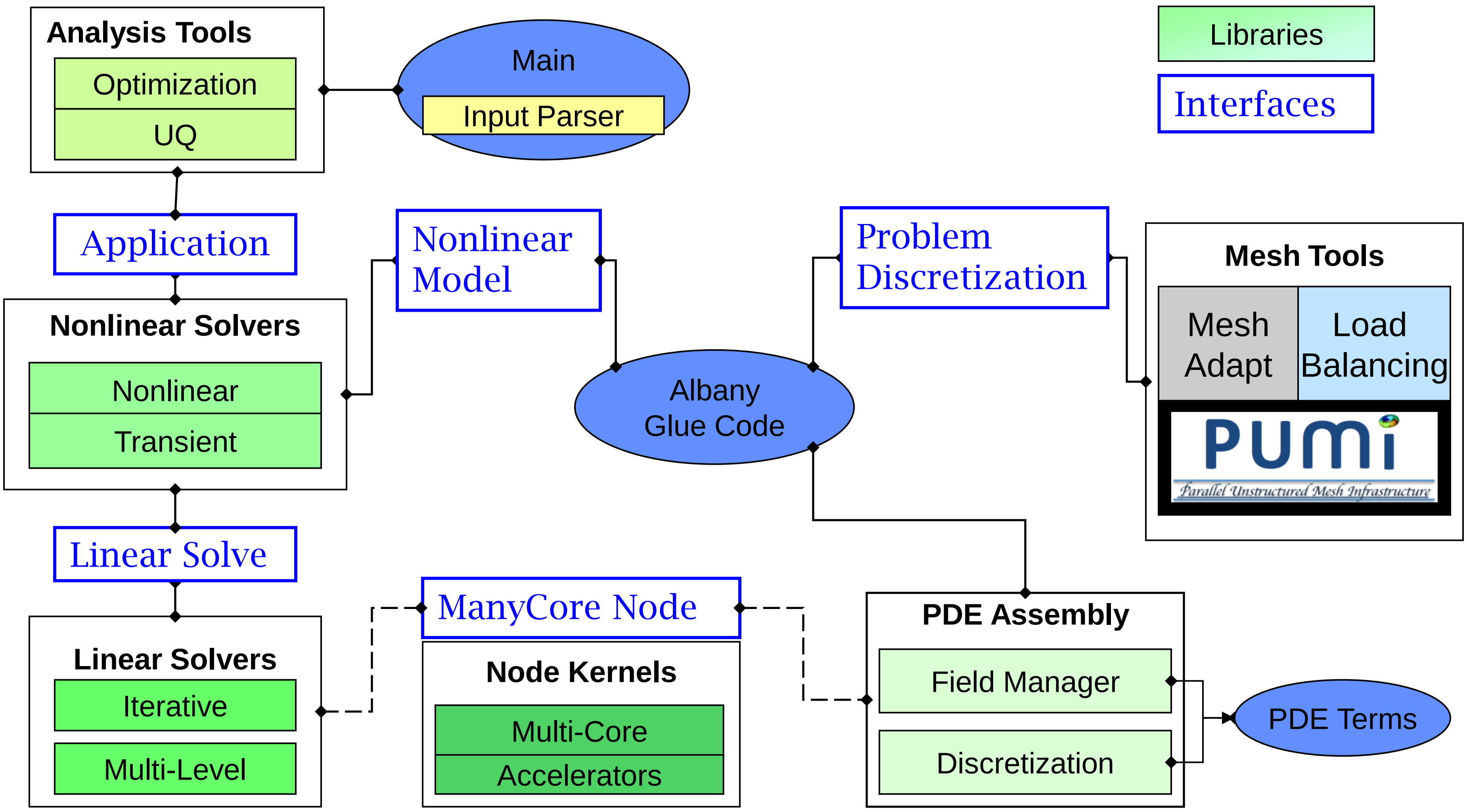


An Extendable Unstructured Mesh Environment

Combination of the FASTMath unstructured mesh technologies with the Albany multiphysics analysis code

- A finite element development environment containing building blocks needed for rapid deployment and prototyping
- A mechanism to drive and demonstrate Agile Components rapid software development vision and use template-based generic programming for the construction of analysis tools
- A Trilinos demonstration application. Albany uses ~98 Sandia packages/libraries.
- Provides an open-source computational mechanics environment and serves as a test-bed for algorithms under development by the Laboratory of Computational Mechanics (LCM) destined for Sandia's production codes

Albany – Agile Component Architecture

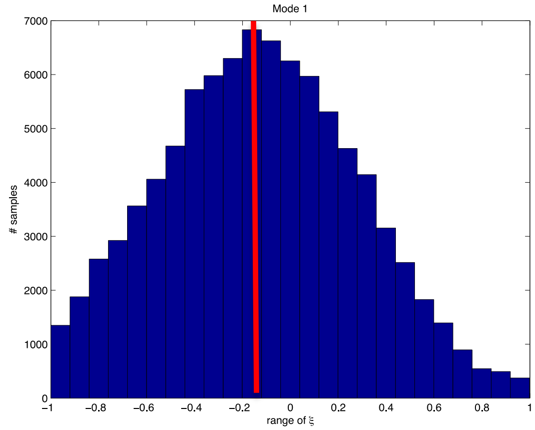


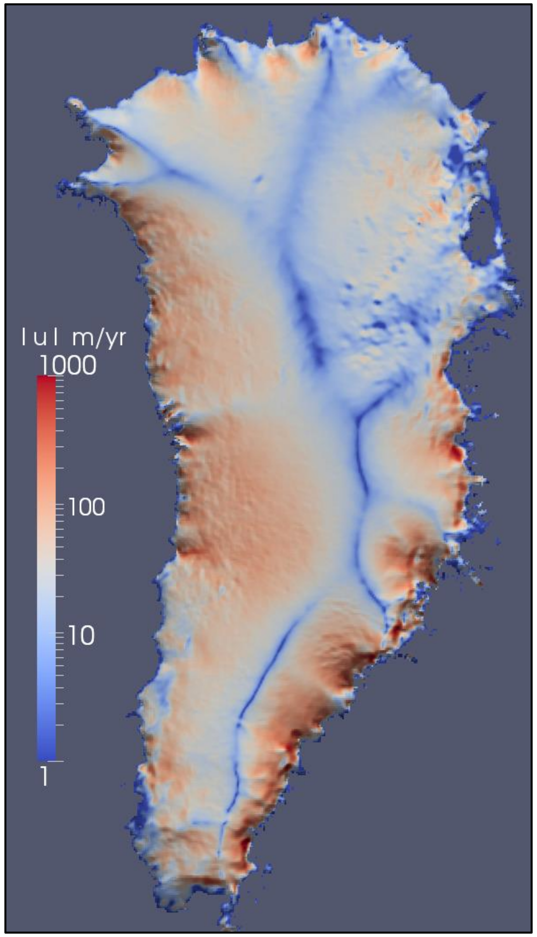
Integrated Technologies Highlight: PISCEES BER SciDAC: 1. Albany/FELIX

Parallel, 3D Unstructured-grid FEM, Implicit, Robust, Verified, Tested, PDE code written ~1 year:

- 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); adjoints in progress
- Same code base as PAALS adaptivity work

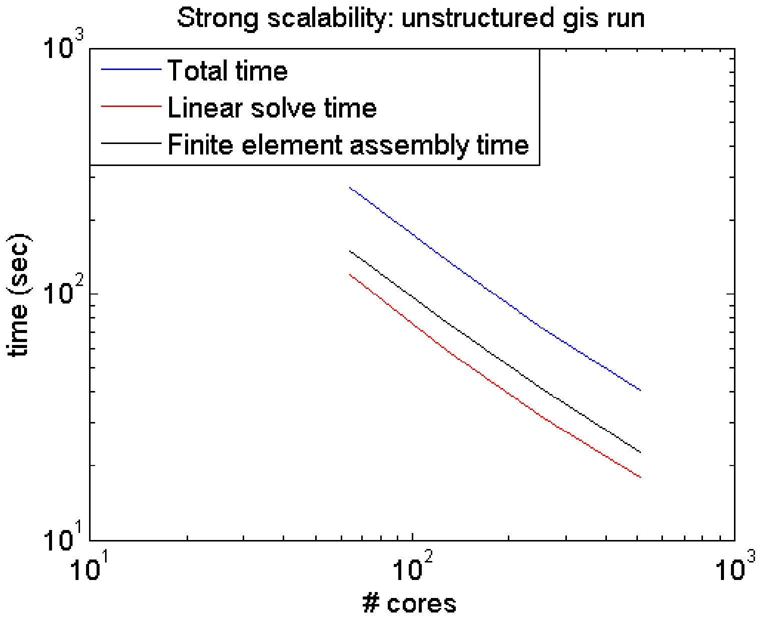
❖ Linked to Dakota
for UQ and Calibration
(KLE → PCE → MCMC)



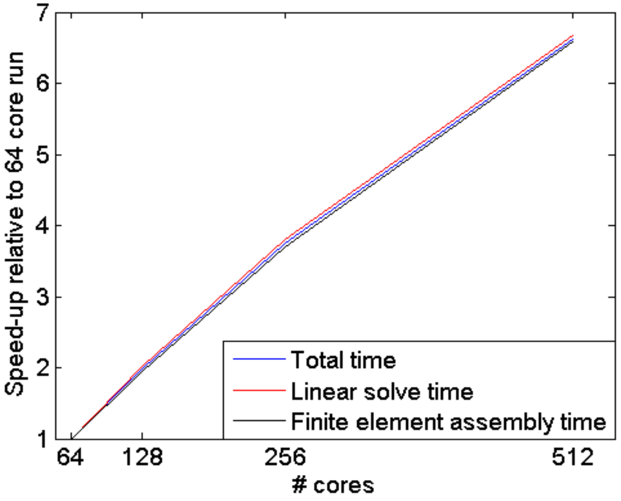


Greenland Ice Sheet
Surface Velocities

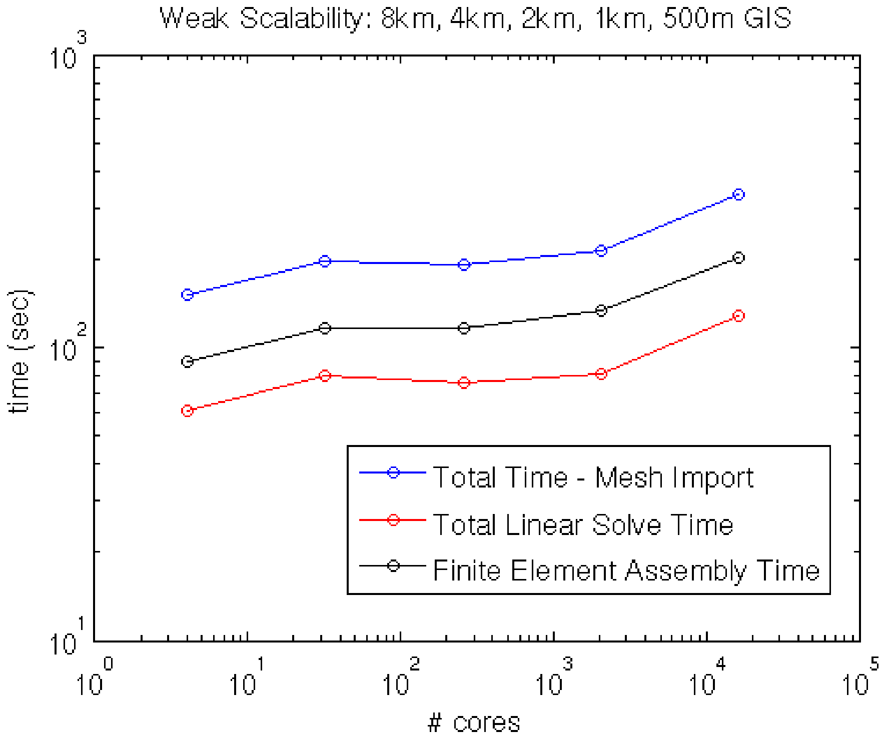
Albany/FELIX Parallel Scaling



Strong scaling



Relative speedup (comparison to 64 core baseline)



Weak scaling: Greenland ice sheet 8km-500m resolution

Courtesy of: Irina Kalashnikova (SNL)

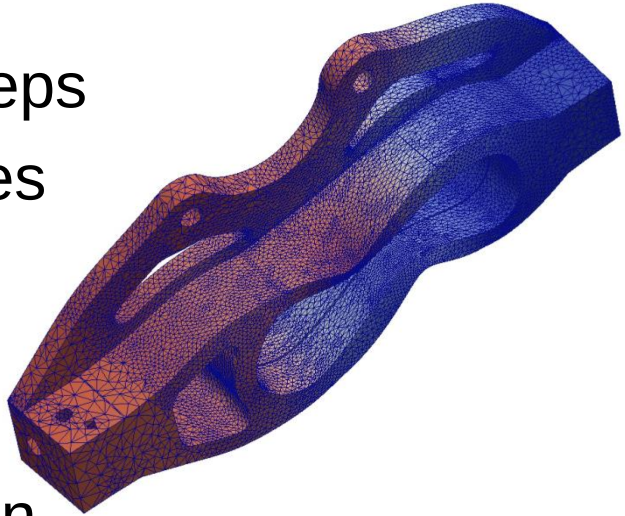
Modeling Large Deformation Structural Failures

Adaptive simulations of finite deformation plasticity with Albany

- Projects include modeling large deformations and weld failures

Efforts on adaptive loops that supports

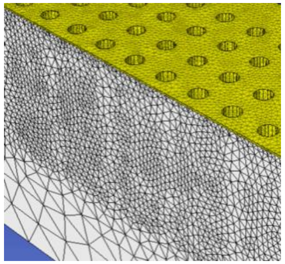
- Solution accuracy via error estimation
    - General error estimation library effort
- High quality element shapes at all load steps
- Accurate solution transfer of state variables
- Predictive load balancing (ParMA, Zoltan) at each adaptive stage
- Expect to add adjoint capabilities for goal oriented error estimation, UQ, Optimization

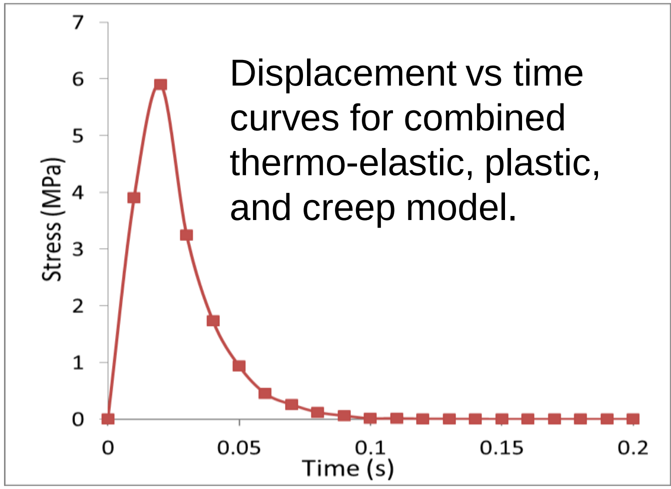


Mechanical Failures in Integrated Circuits

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

- Multi-layer nature of chips interacts with temperature swings, creep, and intrinsic stress of films
- Intrinsic stress in film deposited onto surface and into features causes macroscopic deflection of wafer
- Creep occurs in solder joints during use, delamination during cool-down
- Developing combined constitutive model of thermoelastic, plastic, and creep contributions in ALBANY





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

FASTMath Unstructured Mesh Hand-On Session

Two tracks:

- Using SIGMA tools to construct mesh and its discretization for solving 2-D Laplacian
- Workflow demonstration using Simmetrix/PUMI/PAALS for parallel adaptive simulations

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

ATPESC 2014

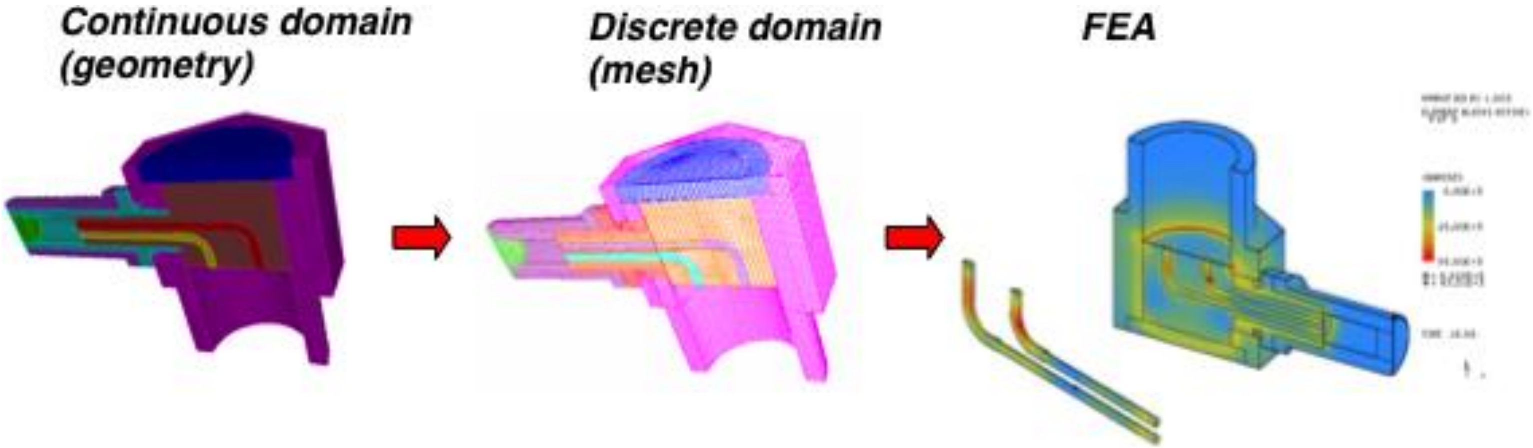
Vijay Mahadevan

FASTMath SciDAC Institute

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 (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 simplifies efficient discretization and solution of PDE on unstructured meshes with FEM.

- To utilize the SIGMA tools effectively, follow workflow to solve a simple 2-D Laplacian on a square mesh (unit cube).
  - 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/atpesc2014

- All other MOAB questions can be directed to

  moab-dev@mcs.anl.gov

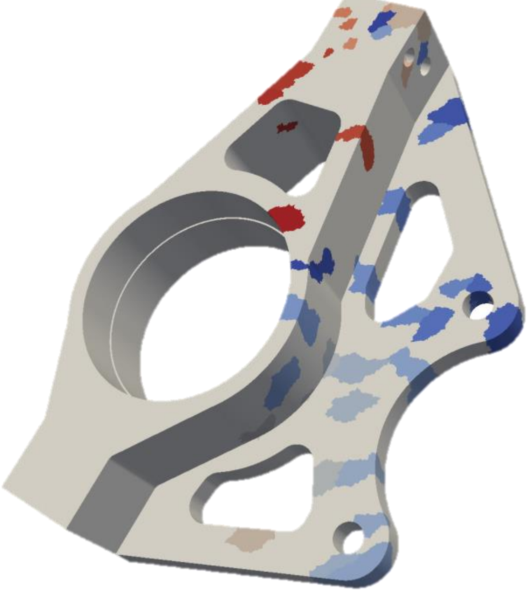
Workflow demonstration using Simmetrix/PUMI/PAALS for parallel adaptive simulations

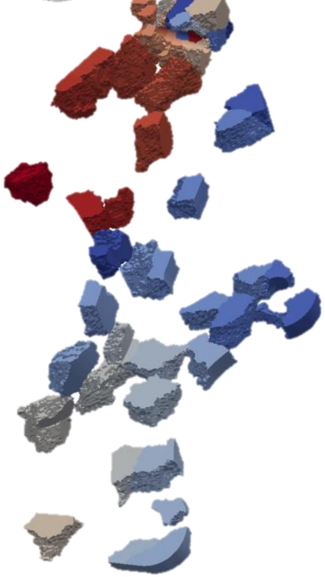
Presenters: Cameron W. Smith and Glen Hansen

FASTMath SciDAC Institute

Hands-on Exercise Outline

- 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
  - ParMetis multi-level graph-based method
  - Partition to 512 parts on 128 cores





Hands-on Exercise Outline

- PAALS
  - In-memory parallel adaptive loop using a plastic deformation model on upright model
  - 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

PAALS

Parallel Albany Adaptive Loop with SCOREC

- **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]