Mesh-based Data and Algorithms across the Simulation Process: anecdotes, activities, and opportunities

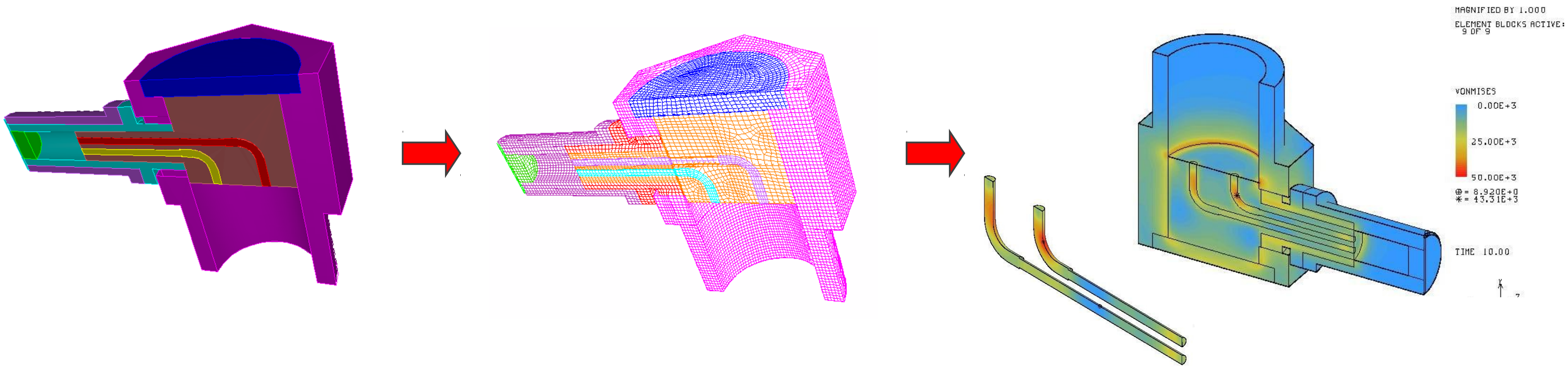
Timothy J. Tautges, Vijay Mahadevan, Rajeev Jain, Tom Peterka
Mathematics and Computer Science Division
Argonne National Laboratory

Joint Lab Workshop
Argonne National Laboratory
November 20, 2012

Outline

- Applications
- Mesh Generation for Reactor Simulation
- Mesh Issues in Coupled Multi-Physics
- Conclusions

Simulation Is Really A Process, Rarely Once-Through



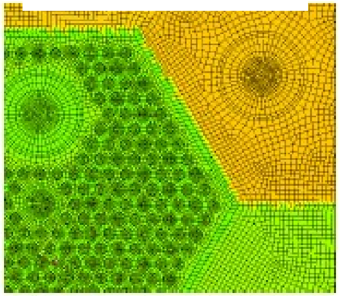
Continuous domain (geometry) ***Discrete domain (mesh)*** ***Simulation*** ***Viz/Analysis***

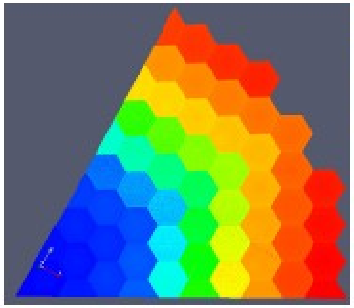
- Spatial domain model the starting point for most PDE-based simulation
- Sometimes geometric details are important, sometimes not
  - MPP-enabled resolution should resolve geometric features (where possible & useful?)
  - The more details you resolve, the harder it is to generate the mesh
- Large-code architecture often organized around handling of the spatial domain (mesh) and fine-grained data on the mesh (fields)

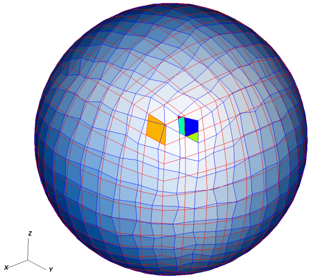
Applications

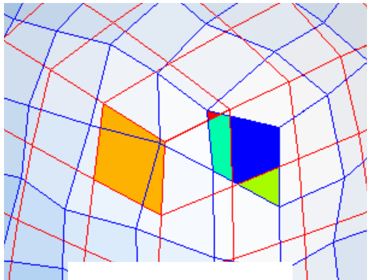
- Reactor simulation
  - Geometry is important
  - Repeated structures sometimes dominant
  - Mostly 3D meshes, some all-hex, some not
- Climate
  - Little/no geometry
  - Mesh usually 2D (+ 1d data vectors for 3rd dimension)
- Fusion
  - Sometimes geometry, sometimes not/little

VHTR Core

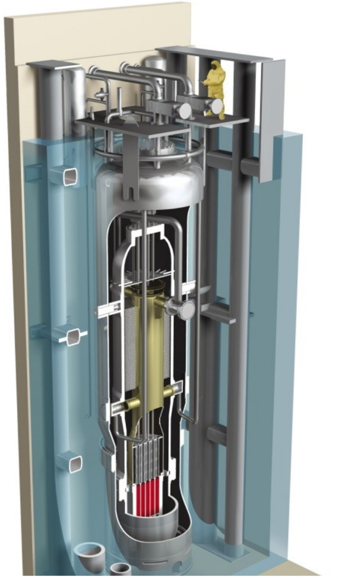




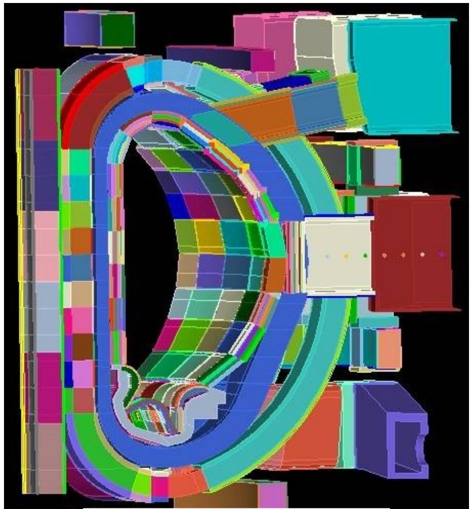




CAM-SE



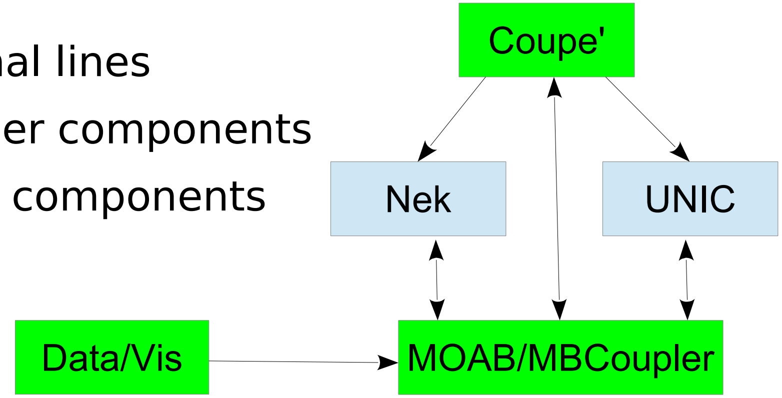
MassLWR Experiment



ITER 40deg

Approach

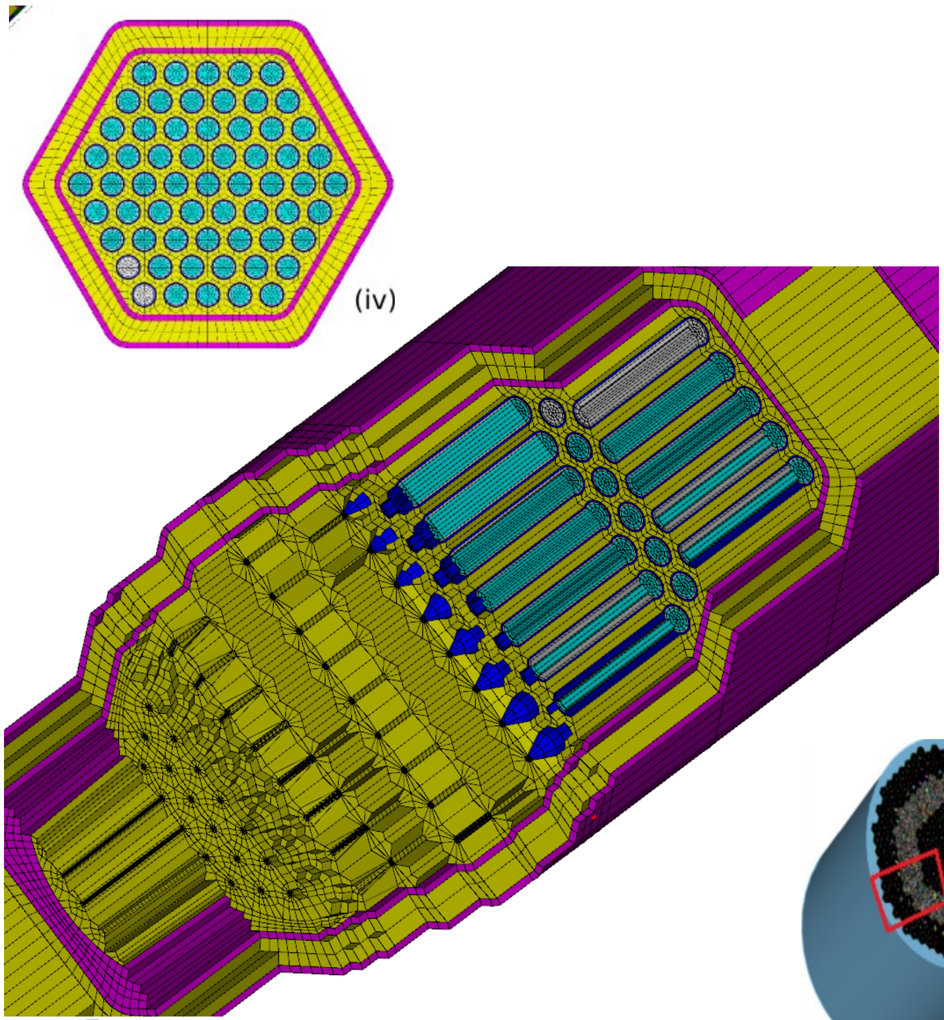
- Small (miniscule)-f framework
  - Distinct components defined along functional lines
  - Individual components can be used w/o other components
  - Applications composed from many of these components
  - Get just what you need, no more

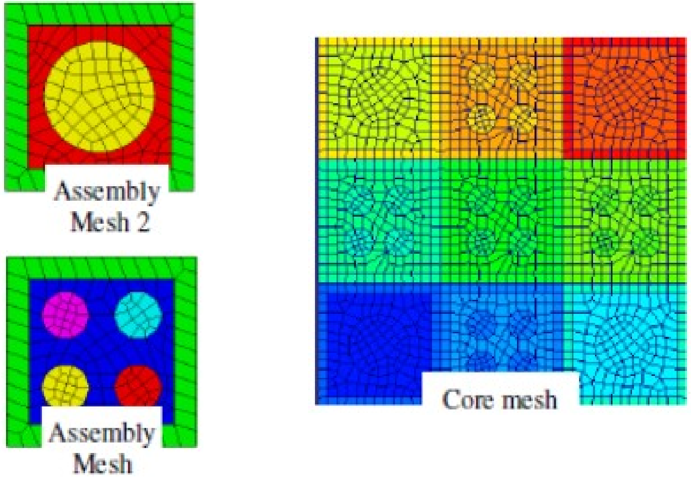


- Mesh-Oriented datABase (MOAB)
  - Library for representing, manipulating structured, unstructured mesh models
  - Supported mesh types:
    - FE zoo (vertices, edges, tri, quad, tet, pyramid, wedge, knife, hex)
    - Polygons/polyhedra
    - Structured mesh
  - Implemented in C++, but uses array-based storage model
  - Mesh I/O from/to various formats (HDF5 native)
  - Parallel representation typical domain-decomposed model, with sharing & ghosting

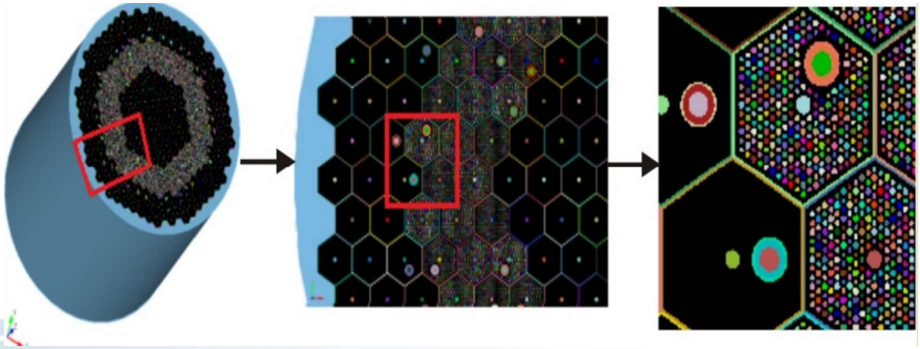
Mesh Generation: 2 Strategies

- Geometry-intensive: CUBIT
- Core lattices: RGG





VHTR full core: 23M hexes, 5.5GB RAM, 30 mins. 313 assemblies.



Coupled Neutron, Fluid, Heat Transport

Code1 (UNIC or proxy)

$$\hat{\Omega}\cdot\vec{\nabla}\Psi + \Sigma_t(E)\Psi = \iint \Sigma_s(E')\Psi(\hat{\Omega}',E')d\hat{\Omega}'dE' + \mathrm{X}(E)\int dE'\,\nu\Sigma_f(E')\int d\hat{\Omega}'\Psi(\hat{\Omega}',E')$$

rho, T-dependent

fission heating

$$\rho C_p u\cdot\nabla T - \nabla\cdot(k\nabla T) = q_{vol} - q_{surf}$$

$$\rho\vec{u}\cdot\nabla\vec{u} = -\nabla p + \frac{1}{\mathrm{Re}}\nabla^2\vec{u}$$

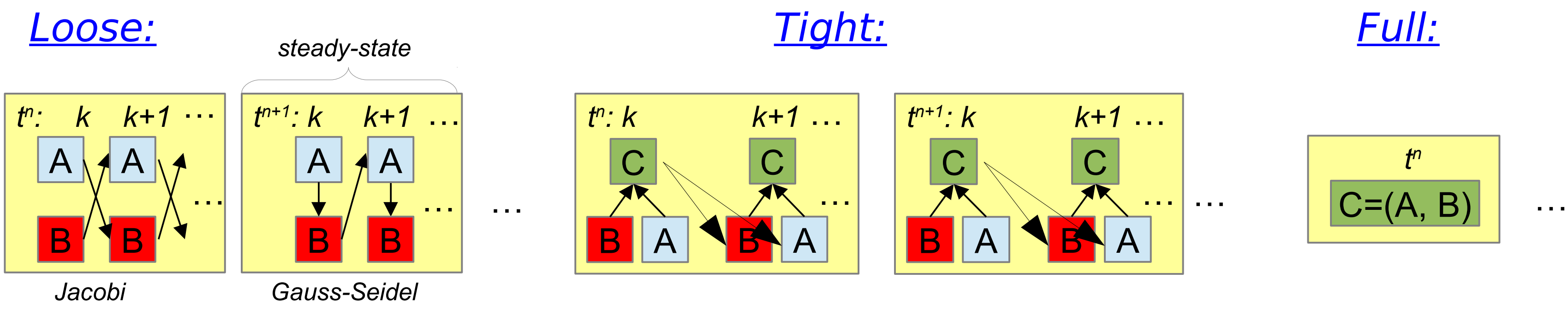
Code 2 (Nek5000 or proxy)

Full/Original Physics Codes

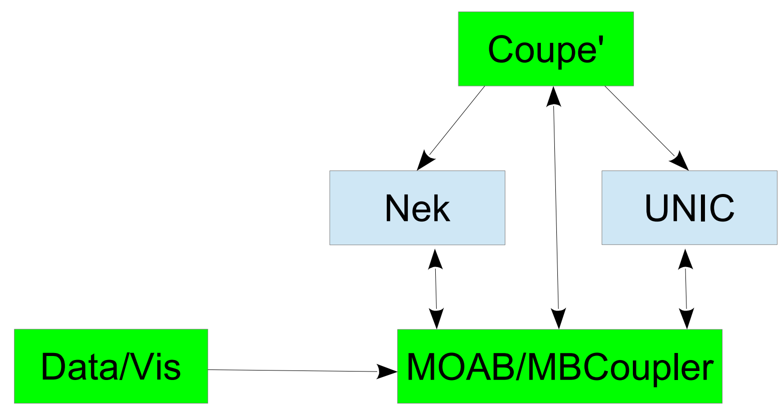
	Nek5000	UNIC
Physics	Incompressible NS	Boltzmann transport
Discretization	SEM w/ LES turb (NxNxN GLL basis)	FEM (linear, quadratic)
Solver	Native semi-implicit with AMG	3-level hierarchy (eigenvalue, energy, space/angle), with PETSc for space/angle
Materials, BCs	User-defined functions	ExodusII-like element blocks, sidesets
Mesh type	Ucd hex	Ucd hex, tet, prism
Implementation	F77 + C, 100k lines	F90, 260k lines
Mesh, data storage	Common blocks	F90 modules
Scalability	2000 Gorden Bell prize, 71% strong scaling on 262k cores	2009 Gordon Bell finalist, 76% strong scaling on 295k cores
Effort invested	~30 man-years	~10 man-years

Coupling Approach

- Different flavors of coupling schemes have variations in stability, accuracy, and software characteristics

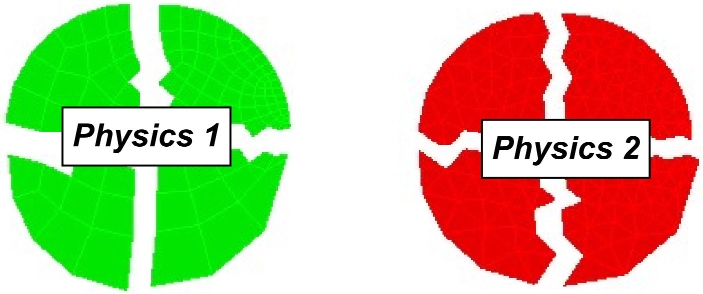


- Driver (Coupe')
  - Support loose, tight coupling with run-time switching
- Use MOAB
  - Solution transfer
  - Other mesh-based services
  - Data conduit

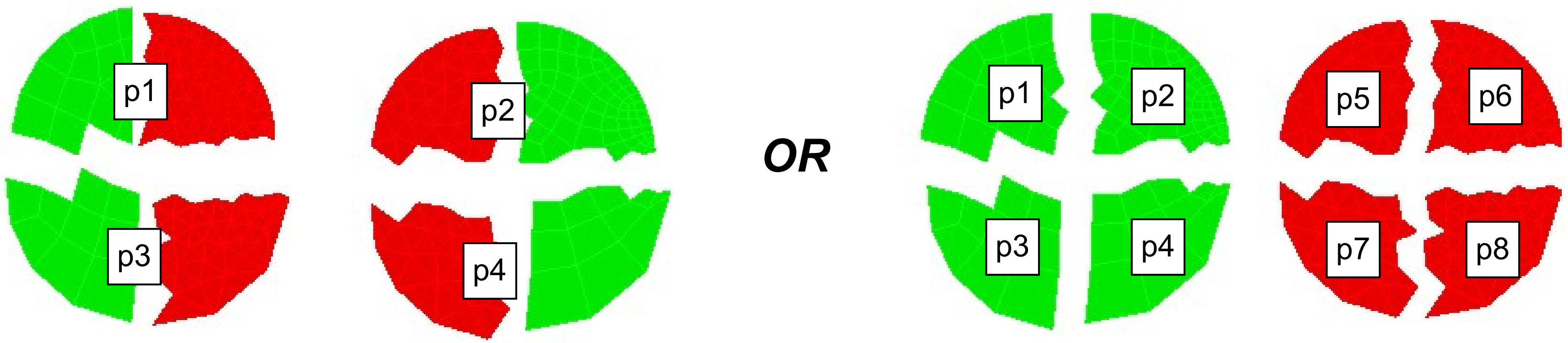


MOAB-Based Solution Transfer

- ***Meshes:*** Each physics type is solved on an ***independent mesh*** whose characteristics (element type, density, etc.) is most appropriate for the physics



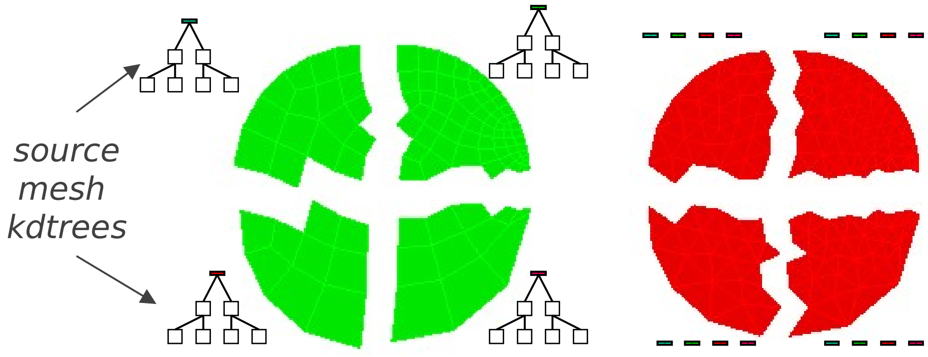
- **Distribution:** Each physics type and mesh is ***distributed independently*** across a set of processors, defined by an MPI communicator for each mesh



- **Implementation:** On a given processor, all meshes are stored in a ***single iMesh instance***, and that instance communicates with all other processors containing pieces of any of those meshes.

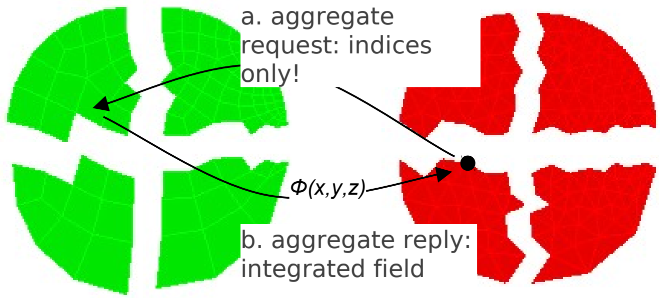
Solution Transfer: 4 Steps

1. Initialization

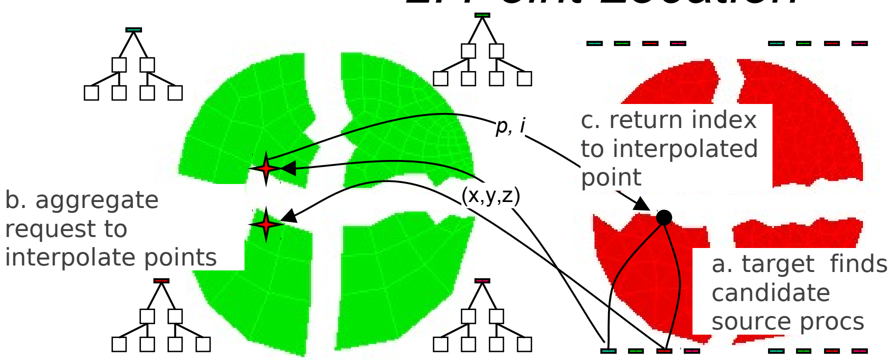


target procs store all kdtree roots

3. Interpolation



2. Point Location



i: (x, y, z), h, (u, v, w)
…

Source proc: index of mapped points: Target position, local element handle, param coords

h, p, i
…

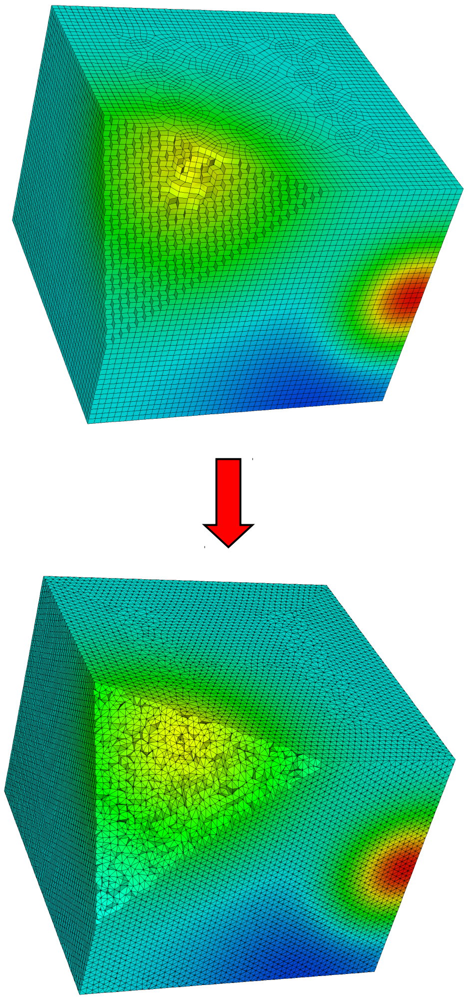
Target proc: local handle, source proc, remote index

4. Normalization

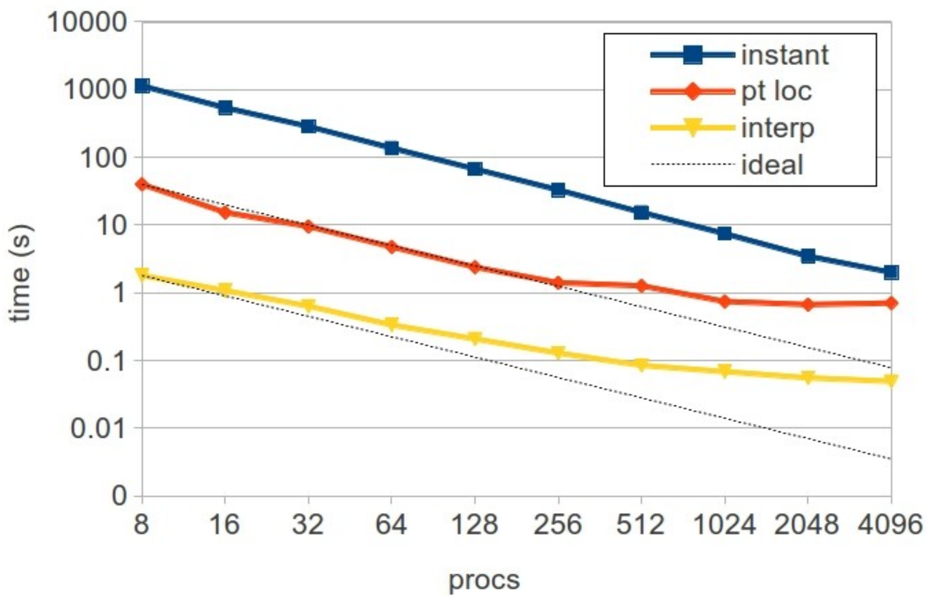
- Minimize data transferred
  - Store index close to source field, communicate indices only
- All communication aggregated, using "crystal router" for generalized all-to-all

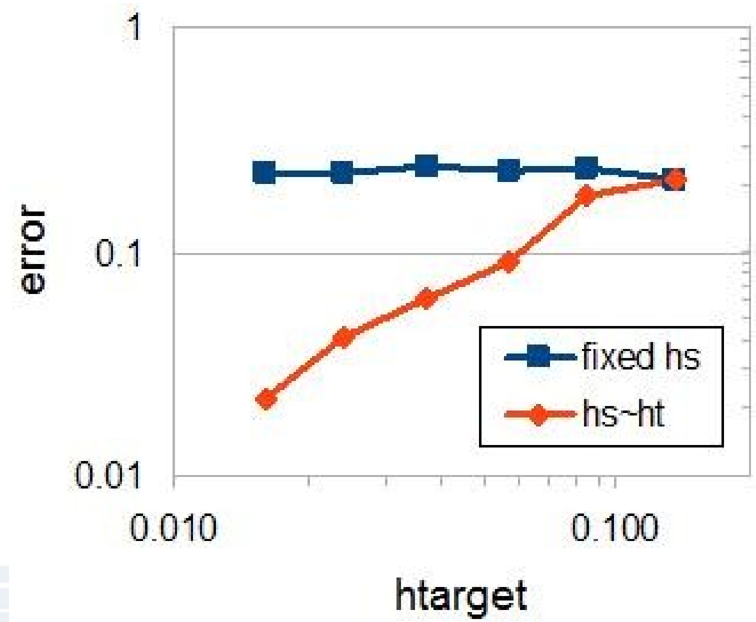
Solution Transfer: Performance, Accuracy

7M Hexes



28M Tets





Exascale Issues

- Partitioning physics over processors
- Parallel solution transfer
- Local tree search
- Memory sharing

Solution Transfer: Distribution Over Processors

- Assuming fixed number of procs and fixed (possibly non-equal) problem sizes for physics, 2 choices for partitioning physics solutions over machine
- Homogeneous: each proc solves a piece of each physics
  - Requires good strong scaling of each physics
  - Can do both Jacobi- and Gauss-Siedel-type loose coupling
  - Easier load balancing, even with sub-cycling in time
- Disjoint: each physics solved on set of procs disjoint from other physics procs
  - Lighter strong scaling requirements
  - Gauss-Siedel scheme leaves processor sets idle, Jacobi requires accurate prediction of runtime
- Our approach: don't over-constrain any of the underlying support (i.e. solution transfer can support both homogeneous and disjoint scenarios)

Solution Transfer: Mesh Search Details

- Current parallel search method does linear search over top-level boxes on each proc, which is both scalability and memory problem
- Change to a rendezvous-type method, where intermediate set of procs with deterministic partition of overall bounding box & intersecting processor boxes directs packets to correct proc(s)
- Local search tree currently a kdtree, but probably more efficient to use a bvh tree
  - Tree search consists of tree traversal (cheap), in-leaf element query (expensive); bvh adds tree complexity to reduce leaf complexity
- In process of implementing/testing bvh tree
- Will implement rendezvous method in FY13

Memory Sharing Between Physics, MOAB

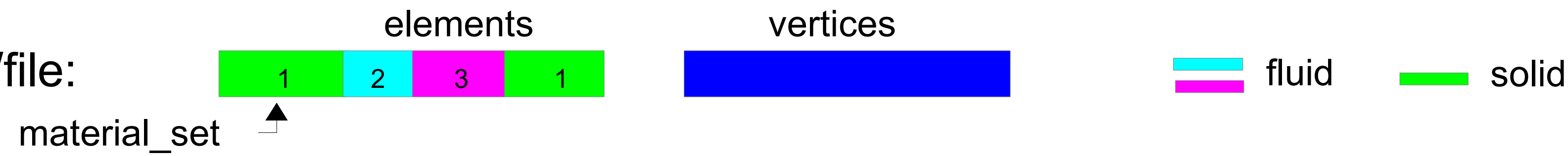
- MOAB uses array-based storage of most “heavy” data, and exposes API functions giving access to contiguous chunks of those data (mesh definition & mesh-based variables)

```
Range::iterator iter = myrange.begin(); int count; double *data;
while (iter != myrange.end()) {
  tag_iterate(tag_handle, iter, myrange.end(), count, (void*&)data_ptr );
  iter += count;
}
```

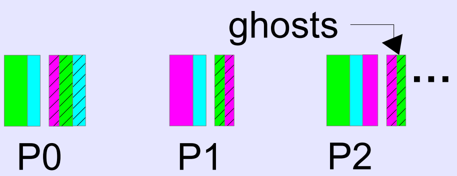
- Small applications show that this almost completely eliminates API cost for accessing variable data memory owned by MOAB
- Advantages:
  - Eliminates memory copy between physics & backplane, saving memory and time
  - Allows direct use of parallel services like I/O, in-situ viz
  - Simplifies workflow (pre, analysis, post) because no issues with data formats for various physics
  - Will allow faster transition to memory manipulations for manycore, GPU
- The fine print: depends heavily on mesh, DOF ordering in physics

Ordering Issues

Serial/file:

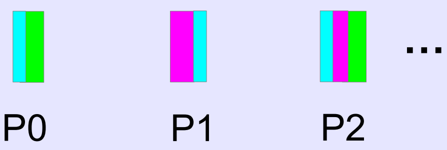


MOAB/Parallel:



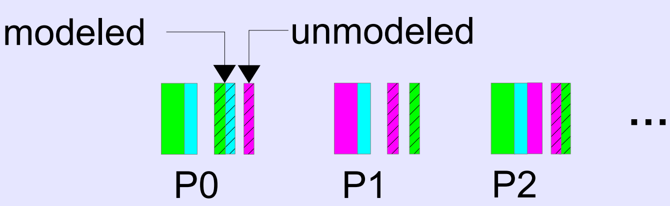
- partition generated w/o regard to material types
- element, set order from file, ghost order from ghost exchange

Nek:



- fluid elements first, then solid elements
- one contiguous index space

UNIC:



- distinguish between “modeled” (has non-ghost elems) and “unmodeled” (no non-ghost elems) ghost materials

- *Moral: to meet application requirements, reordering often necessary, either during handoff to physics, or in MOAB before handoff*

Opportunities

- Mesh generation
  - AMR
- Coupled multi-physics
  - More physics codes (Saturne? Code Aster?)
  - Solution transfer scalability
- Partitioning/reordering
  - Multiple ordering criteria, e.g. by proc then material