PUMI

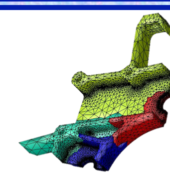
Parallel Unstructured Mesh Infrastructure

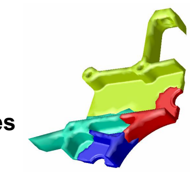
Parallel Unstructured Mesh Infrastructure for Massively Parallel Adaptive Simulation

Seegyoung Seol, Daniel Ibanez,
Cameron Smith and Mark S. Shephard
Scientific Computation Research Center
Rensselaer Polytechnic Institute

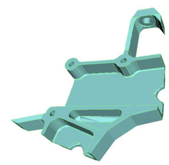
SCOREC

PUMI





Mesh
0-3D topological entities and adjacencies



Geometric model
analysis domain

Fields
distribution of solution over mesh

Partition model
distribution of mesh across computing resources

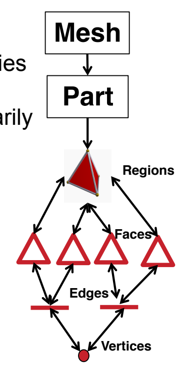
Parallel Control
communication utilities

SCOREC

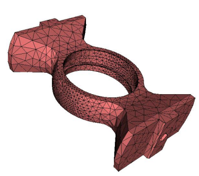
Background

Geometry-Based Analysis

- Geometry, Attribute: analysis domain
- Mesh: 0-3D topological entities and adjacencies
- Field: distribution of solution over mesh
- Common requirements: data traversal, arbitrarily attachable user data, data grouping, etc.
- Complete representation: store sufficient entities and adjacencies to get any adjacency in O(1) time



Geometric model



Mesh

SCOREC

Notations

- P_i: distributed mesh part mapped to a process (1-to-N)
- V_i^d: entity of dimension *d*, where V is G (geometric model), M (mesh) or P (partition model)
  E.g. M_i^3: mesh regions, G_i^2: geometric face , P_i^0: partition vertex
- $\mathcal{P}[V_i^d]$: operator to return a set of part id's where V_i^d exists
  - *E.g.* $\mathcal{P}[M_1^0] = \{P_0, P_1, P_2\}$
- $V_i^d \sqsubset W_j^q$, $d \le q$: classification which represents a relation from V_i^d to W_j^q, *where* V_i^d *is partial representation of* W_j^q
  - $M_i^d \sqsubset G_j^q$, $d \le q$: geometric classification
  - $M_i^d \sqsubset P_j^q$, $d \le q$: partition classification
- S_i : a set of entities

SCOREC

General Mesh Data Structure

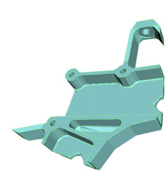
Functional Requirements for Adaptive Simulations

- *Part*: A unit of mesh data decomposition for distribution on parallel computers.
- *Mesh entities*: a constituent of mesh distinguished by type
  - vertex (0D), edge (1D), face (2D), or region (3D)
- *Adjacencies*: how the mesh entities connect to each other.
- *Geometric classification*: a relation that each mesh entity maintains to a geometric model entity for partial representation
- *Entity set*: mechanism for grouping mesh entities
- *Tag*: mechanism to attach arbitrary user data (tag data) to a part, entity set or mesh entity
- *Iterator*: mechanism to traverse mesh entities in a specific range with various options (type, classification, etc.)

SCOREC

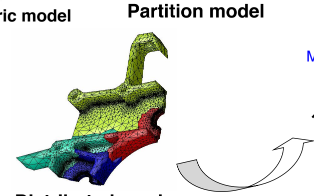
Distributed Mesh Data Structure

- Capability to partition mesh to multiple parts per process

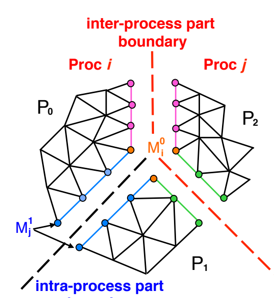


Geometric model

Partition model



Distributed mesh



SCOREC

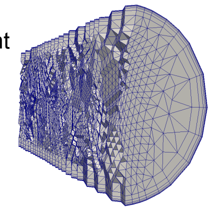
Distributed Mesh Requirement (1/2)

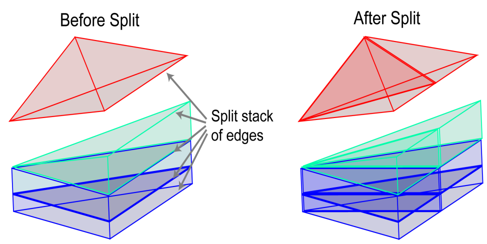
- File I/O – parallel mesh/set/tag loading, saving
- Part – create, delete, query (neighbor, id, etc), entity iterator
- Part boundary – query, entity iterator
- Entity – query (owner part, status, copies, etc.)
  - Entity ownership imbues the right to modify (in other words, only owning part can modify the entity and transfer the modification to its remote copies)
- Modification
  - Migrating entities and p-sets to destination part
  - Pulling part boundary entity's *remote* partition objects to the owner part
  - Pushing owner's vertex coordinates to remote copies
  - Ghosting – temporarily keeping remote adjacent entities on local part
  - Tag –automatic tag data migration during migration
- Mesh partitioning control
  - Static/dynamic, global/local
  - Weight control per individual entity/set or type/topology

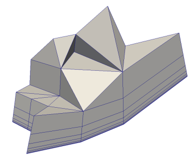
Distributed Mesh Requirement (2/2)

Boundary layer stacks

- NP-set: Entity set w/o single part constraint
- P-set: Entity set with single part constraint
  - Data structure for boundary layer stack
  - Entity is partition object entity
  - Entity can be contained in at most one p-set

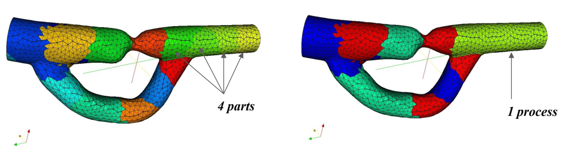






Distributed Mesh Representation (1/6)

- $Part_i$ consists of mesh entities assigned to i^{th} part
- Multiple-parts per process
  - Changing number of parts per process
  - Dealing with problems with current graph-based partitioners that tend to fail on really large numbers of processors (*See Slides 19-21*)
  - Architecture-aware two-level mesh partitioning (*See Slides 22-23*)
  - For effective manipulation, a mesh instance defined on each processor contains part handles assigned to the process



A 3D mesh in 4 parts per process (16 parts total)

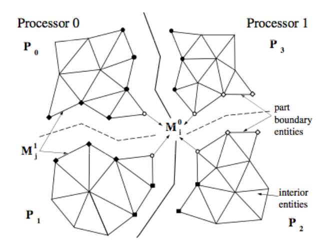
(LEFT) Different color represents different part

(RIGHT) Different color represents different process

Distributed Mesh Representation (2/6)

Each part P_i assigned to a processor

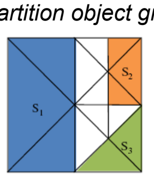
- Uniquely identified by handle or global ID
- Treated as a serial mesh with the addition of *part boundaries*
  - *Part boundary*: groups of mesh entities on shared links between parts
  - *Part boundary entity*: duplicated entities on all parts for which they bound with other higher order mesh entities
  - *Remote copy*: duplicated entity copy on non-local part
  - *Partition object* : basic unit to assign dest. part id in migration
  - *Residence parts and owing part* : list of parts where the entity exists and the part designated to be in charge of modification



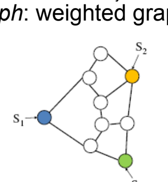
Distributed Mesh Representation (3/6)

Partition Object

- Basic unit to assign destination part id in mesh migration
  - p-set
  - mesh entity with no higher order adjacency *not contained in* p-set
- For partition object x, residence part operator $\mathcal{P}(x)$ returns a set of part id's where x exists based on adjacencies.
  E.g. $\mathcal{P}[M_i^0] = \{P_0, P_1, P_2, P_3\}$, $\mathcal{P}[M_j^1] = \{P_0, P_1\}$
- *Partition object graph*: weighted graph G(V, E)



A mesh part with 3 p-sets



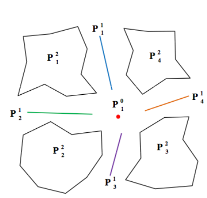
Partition object graph

- **Node *V*: partition object**
- **Edge *E*: dependencies between graph nodes identified by adjacencies**
- **Node and edge weights**

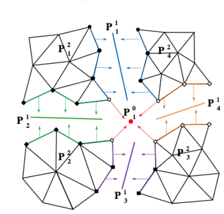
Distributed Mesh Representation (4/6)

Partition Model

- a conceptual model existing between a geometric model and distributed mesh representing mesh partitioning in topology
- Partition (model) entity: a topological entity in the partition model, P_i^d, representing a group of mesh entities of dimension *d with* the same *residence parts*.



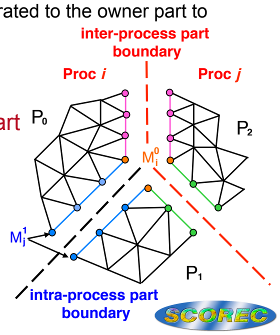
Partition model of mesh in Slide 14



Partition classification in arrows

Distributed Mesh Representation (5/6)

- For each partition model entity, owning part is *defined* by a *rule* which says the part with the *fewest* number of *partition object* entities is the owner.
  - Rationale: keeping load balance during adaptation - part boundary entity's *remote* partition objects are migrated to the owner part to obtain cavity
- Proper maintenance of partition classification is all about modifying entity's residence parts and owning part
  - $M^1_j@P_0$ and $M^1_j@P_1$ know they are duplicated in P_0 and P_1
  - M^1_j's owning part changes dynamically as mesh partitioning changes



Distributed Mesh Representation (6/6)

Other tools for efficient mesh modification

- No global ID synchronization
  - global part ID = process rank * local part ID
    local part ID = 0..n-1, where n is # parts per process
  - No global entity ID
- No mesh size operation
  - O(1) Entity search based on adjacencies
  - O(N) Mesh migration, N - # entities to migrate
- Parallel Control Utility
  - Provides parallel infrastructure to control communications
  - Being extended to deal with hybrid (message passing and threads)
  - More on this later

Mesh Migration (1/2)

Purpose: Moving mesh entities between parts

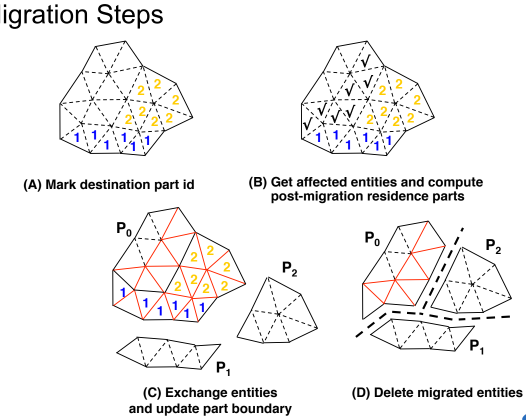
- Dictated by operation - in swap and collapse it's the mesh entities on other parts needed to complete the mesh modification cavity
- Entities to migrate are determined based on adjacencies

Major Complexities

- A function of mesh representation w.r.t. adjacencies, p- set and arbitrary user data attached to them
  - Complete mesh representation can provide any adjacency without mesh traversal - a requirement for satisfactory efficiency
- Performance factorized by
  - Synchronization, communications, load balance and scalability
  - How to benefit from on-node thread communication (all threads in a processor share the same memory address space)

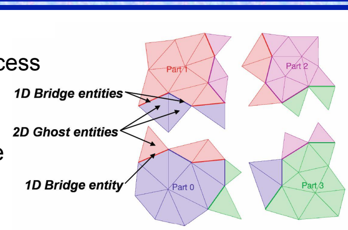
Mesh Migration (2/2)

Migration Steps



Ghosting (1/2)

- *Goals*: localizing off-part mesh data to avoid inter-process communications for computations
- *Ghost*: read-only, duplicate entity copies not on part boundary including tag data



- *Ghosting rule*: triplet (ghost dim, bridge dim, # layers)
  - Ghost dim: entity dimension to be ghosted
  - bridge dim: entity dimension used to obtain entities to be ghosted through adjacency
  - # layers: the number of ghost 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, use ghost_dim=3, bridge_dim=2, and # layers=2

(source: FASTMath iMeshP.h)

Ghosting (2/2)

Ghosting Steps