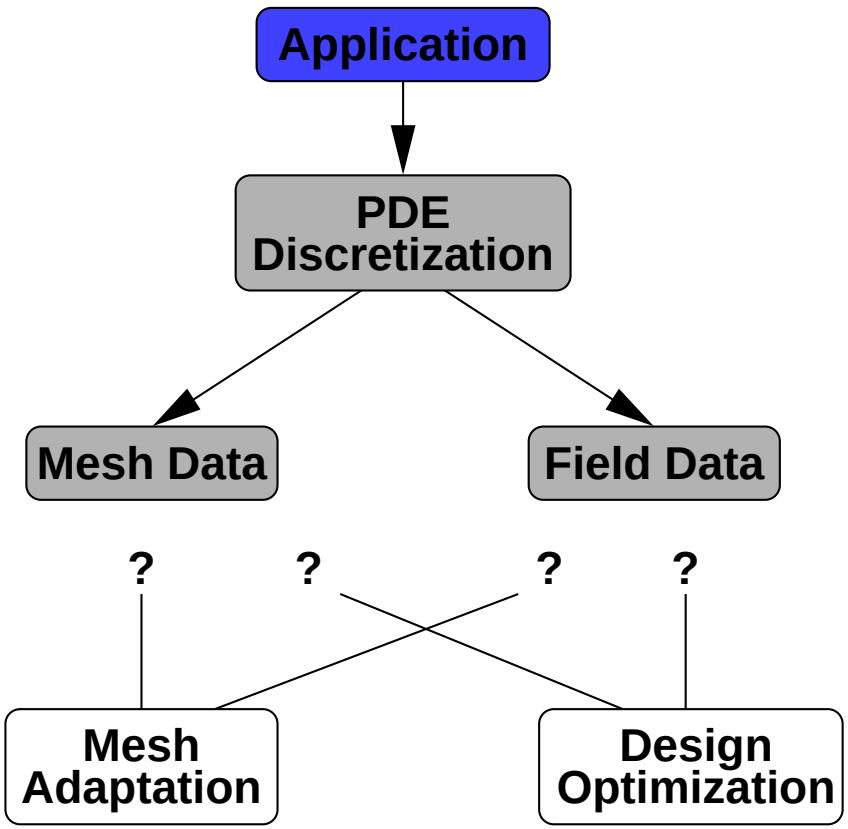
The ITAPS Mesh Interface

Carl Ollivier-Gooch

Advanced Numerical Simulation Laboratory, University of British Columbia

Needs and Challenges for Unstructured Mesh Usage



Needs and Challenges for Unstructured Mesh Usage

PDE-based Applications Need Services Like...

- Mesh/geometry access
- Solution field data storage / manipulation
- Mesh transformations
- Adaptive refinement
- Design optimization

Needs and Challenges for Unstructured Mesh Usage

PDE-based Applications Need Services Like...

- Mesh/geometry access
- Solution field data storage / manipulation
- Mesh transformations
- Adaptive refinement
- Design optimization

Re-use of Existing Code is Hard Because...

- Sometimes bound to a particular framework
- Nearly always data structure-specific
- Libraries have incompatible interfaces
- Cross-language issues

Goals of the ITAPS Effort

ITAPS = Interoperable Tools for Advanced Petascale Simulation

To provide interoperable support for the mesh operations needed by real large scale scientific computing applications

- Interoperability
  - Data structure neutrality
  - Standard API
- Flexibility: different apps represent the same data in different ways
- Efficiency = low overhead + low programmer impact
- Support for new services on top of existing applications

The Component Paradigm

The Idea

- Standardize the data model
- Create common interfaces
- Use existing tools for language issues

The Component Paradigm

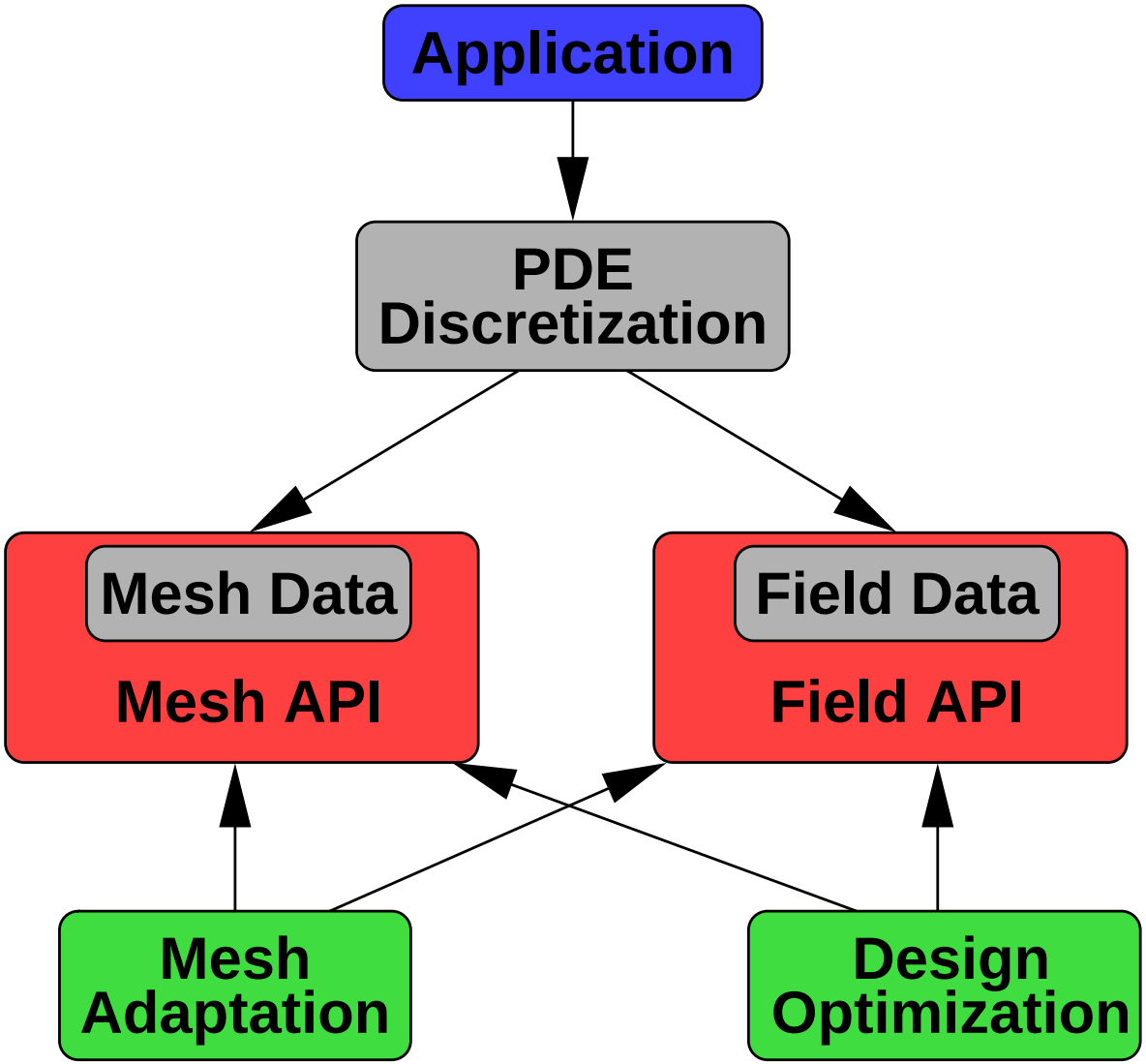
The Idea

- Standardize the data model
- Create common interfaces
- Use existing tools for language issues

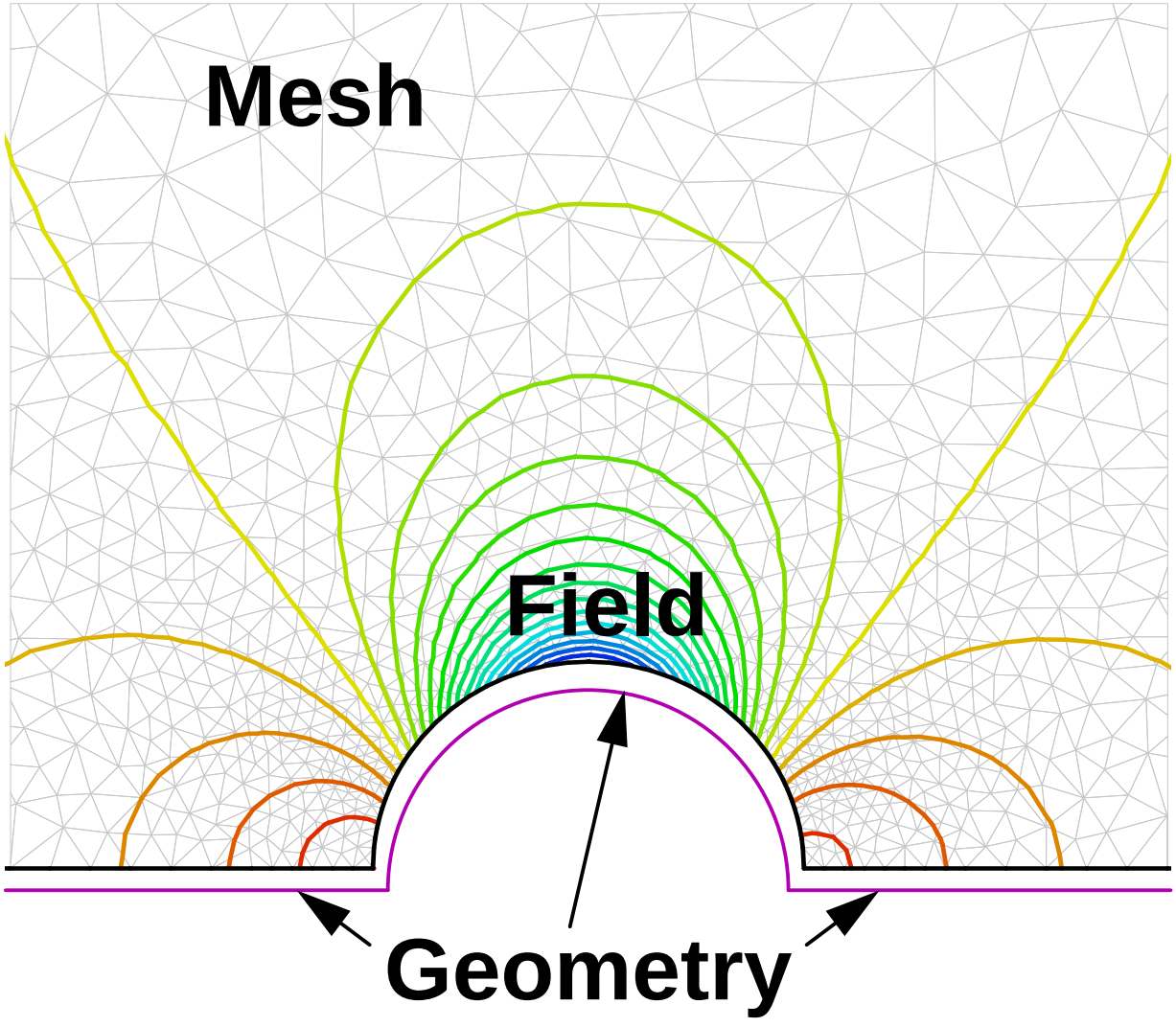
Interoperability Benefits

- Horizontal: allow applications to choose implementation of a given service which works best for them
- Vertical: choose collection of services which interoperate through common interfaces to build higher-level application

The Component Paradigm

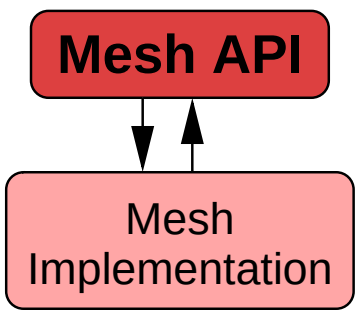


Scope of ITAPS Components



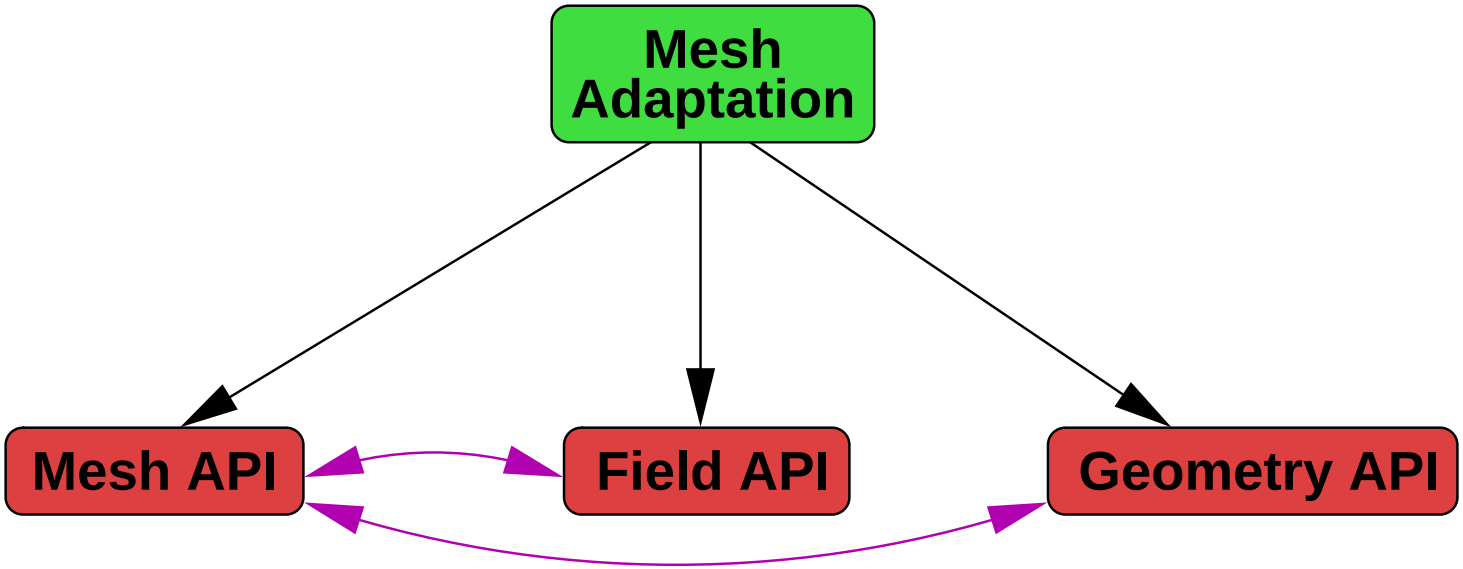
Usage Modes for ITAPS Component API

Implementation



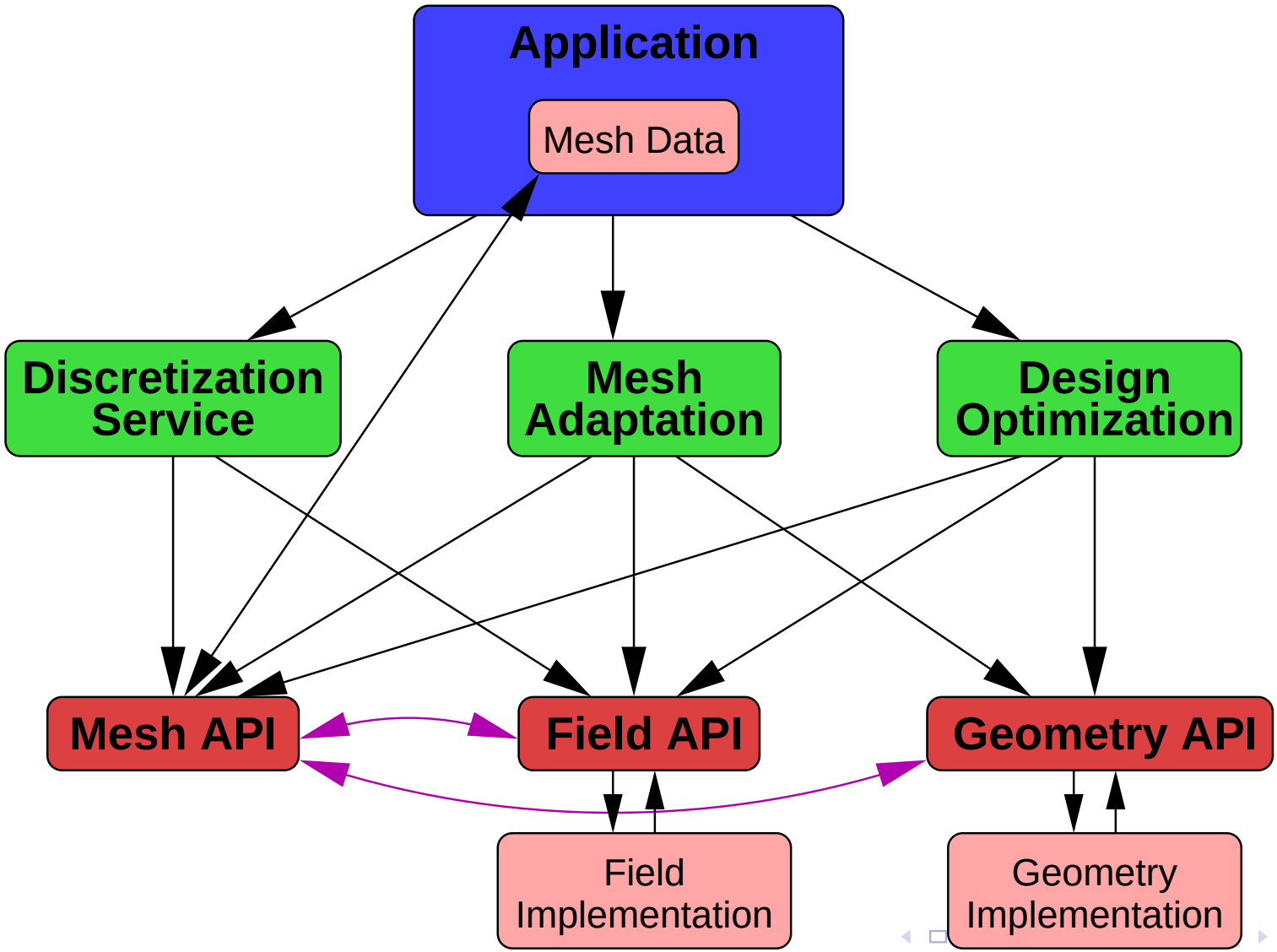
Usage Modes for ITAPS Component API

Service



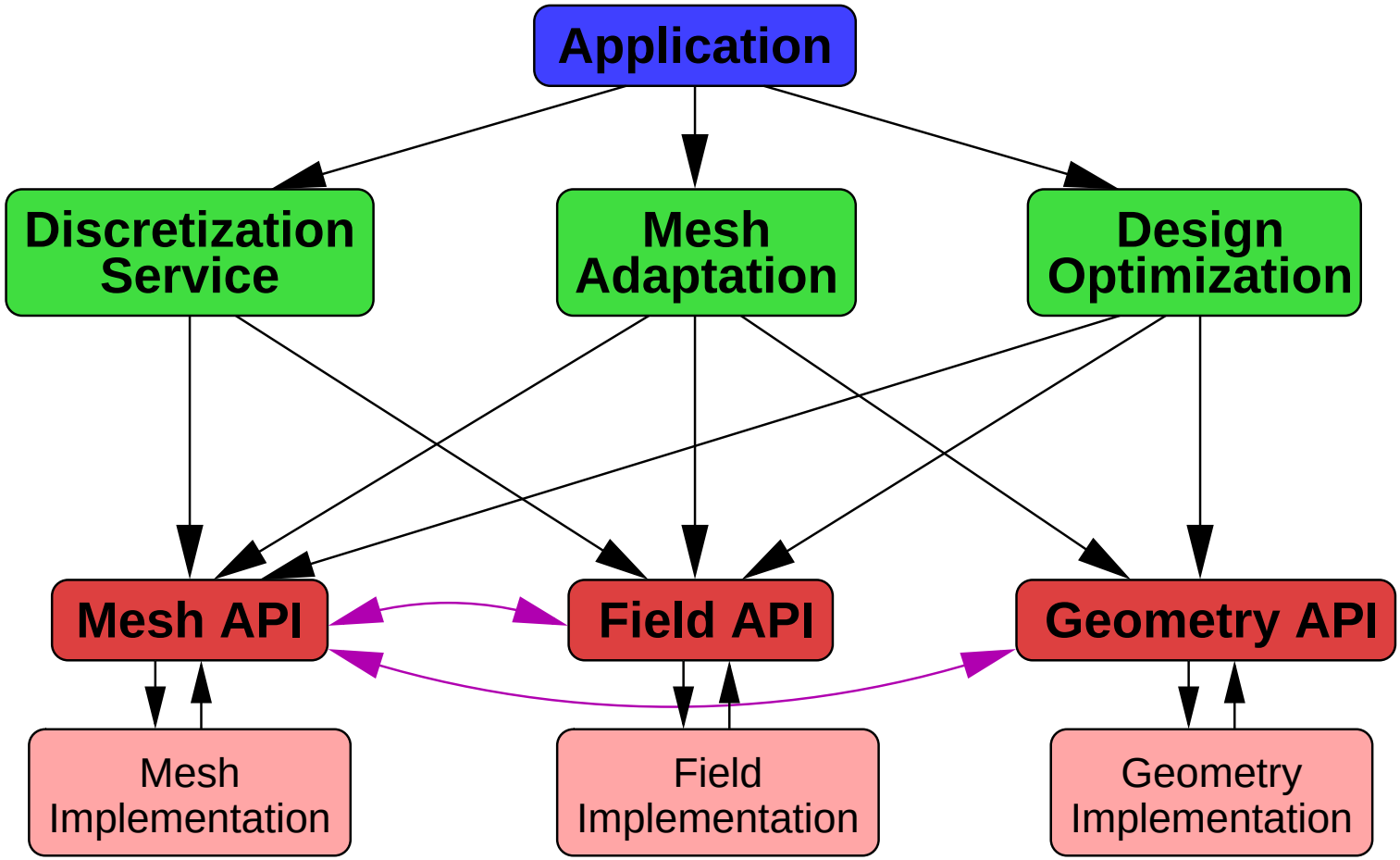
Usage Modes for ITAPS Component API

Application with Embedded (Partial?) Implementation



Usage Modes for ITAPS Component API

Assembled Application



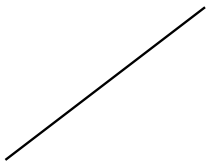
ITAPS Data Model

Entities

Vertex

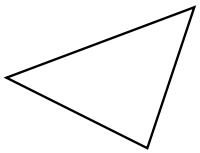
Point

Edge

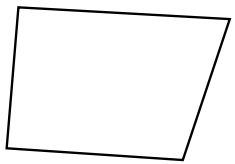


Line Segment

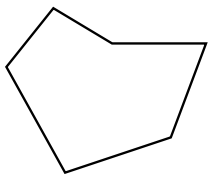
Face



Triangle

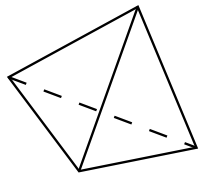


Quadrilateral

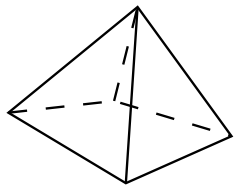


Polygon

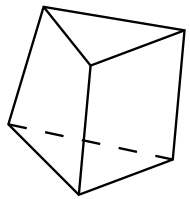
Region



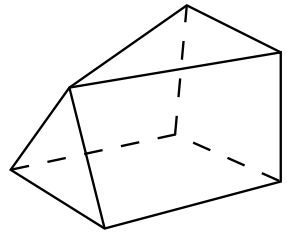
Tetrahedron



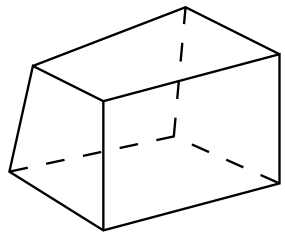
Pyramid



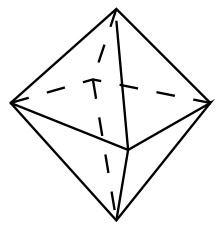
Prism



Septahedron



Hexahedron

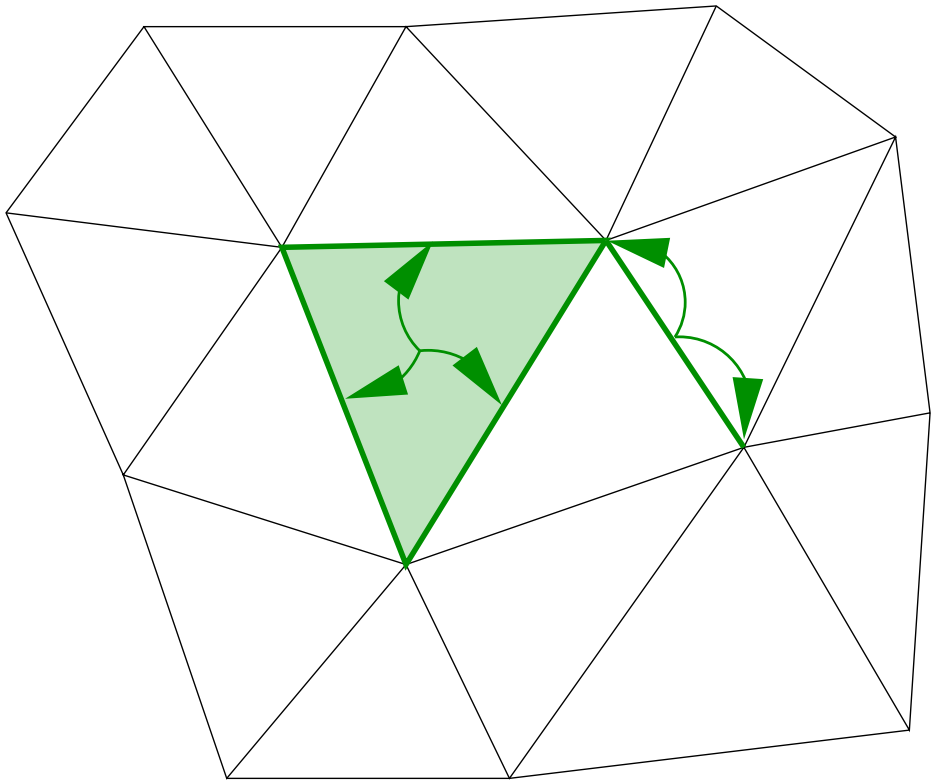


Polyhedron

ITAPS Data Model

Entities

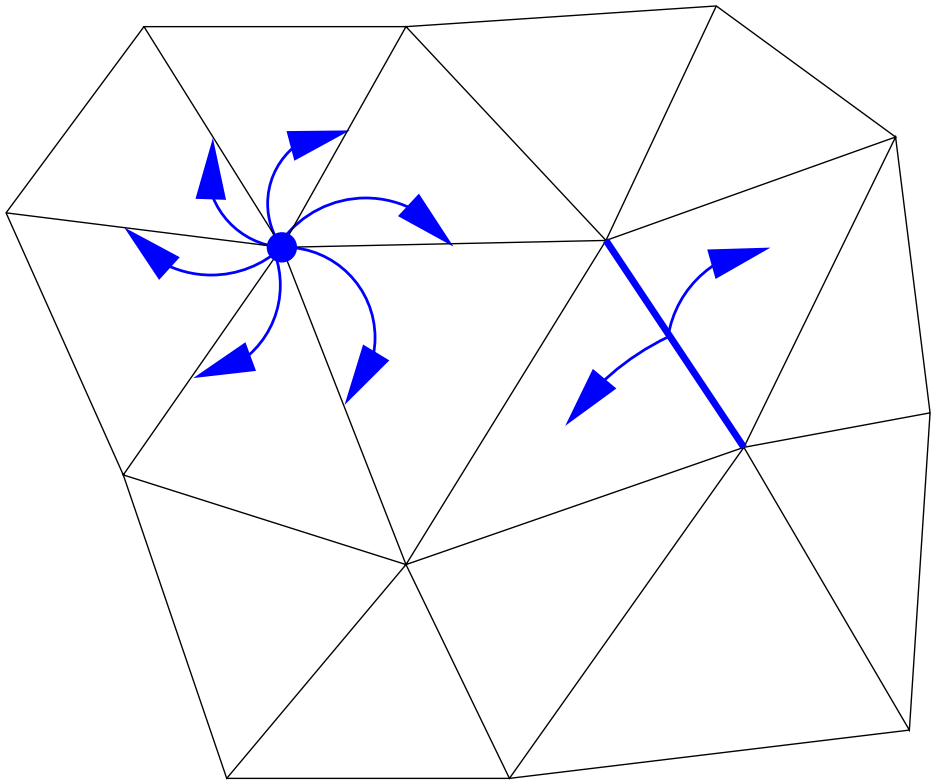
- Downward Adjacency
  - From entity to entities on its closure
- Upward Adjacency
  - From entity A to entities that have A on their closure
- Second Adjacencies
  - From entity A through adjacent entities of type *p* to *their* adjacent entities of type *q*



ITAPS Data Model

Entities

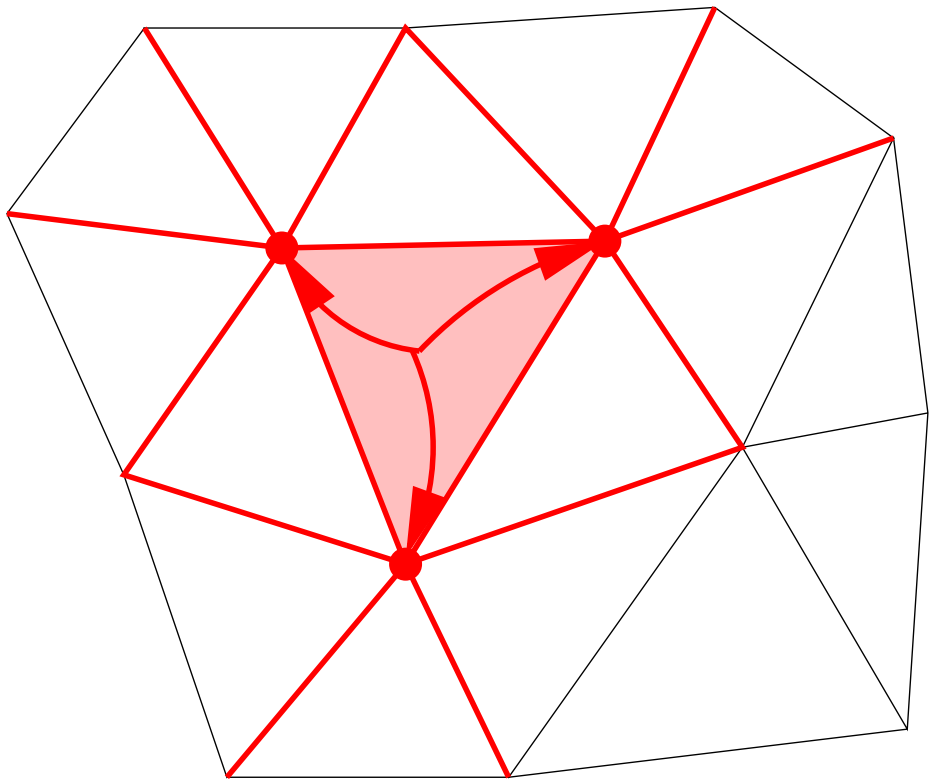
- Downward Adjacency
  - From entity to entities on its closure
- Upward Adjacency
  - From entity A to entities that have A on their closure
- Second Adjacencies
  - From entity A through adjacent entities of type *p* to *their* adjacent entities of type *q*



ITAPS Data Model

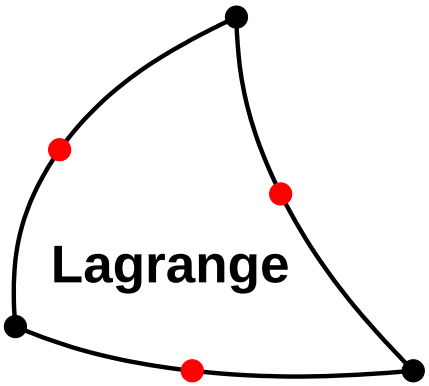
Entities

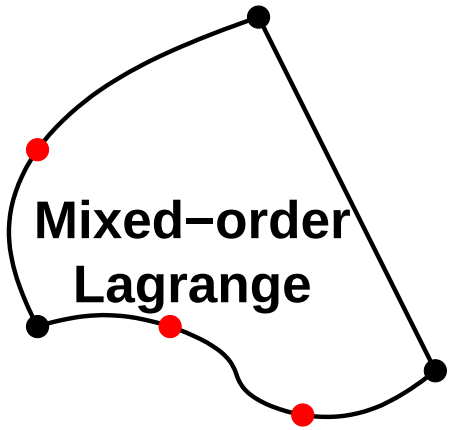
- Downward Adjacency
  - From entity to entities on its closure
- Upward Adjacency
  - From entity A to entities that have A on their closure
- Second Adjacencies
  - From entity A through adjacent entities of type *p* to *their* adjacent entities of type *q*

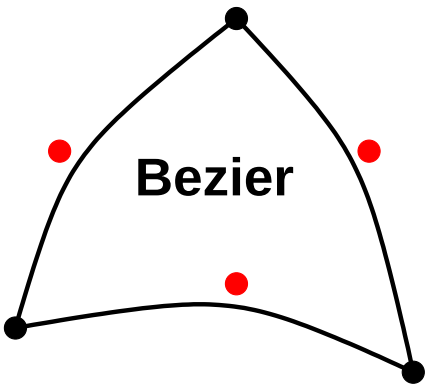


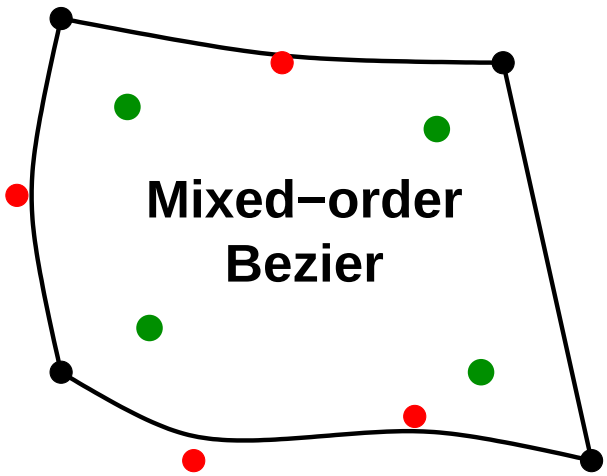
Entities

Entity Shape









ITAPS Data Model

Entity Sets

Definition

Entity set is an arbitrary collection of entities.

- Unordered, unique collection (true set)
- Ordered, non-unique collection (list)

Examples

- Faces sharing a common boundary condition
- Regions on a single processor in a parallel computation
- All mesh entities in one mesh in a multigrid sequence

ITAPS Data Model

Entity Sets

Definition

Entity set is an arbitrary collection of entities.

- Unordered, unique collection (true set)
- Ordered, non-unique collection (list)

Examples

- Faces sharing a common boundary condition
- Regions on a single processor in a parallel computation
- All mesh entities in one mesh in a multigrid sequence

Relationships Between Entity Sets

- Membership (element or subset) or Parent-child (heirarchical)

ITAPS Data Model

Tags

Definition

Tags are used to associate arbitrary data with an entity or entity set.

Examples

- Boundary condition or material specification
- Classification onto geometry
- Local mesh length scale

ITAPS Data Model

Tags

Definition

Tags are used to associate arbitrary data with an entity or entity set.

Examples

- Boundary condition or material specification
- Classification onto geometry
- Local mesh length scale

Supported Types of Tag Data

- Typed tags: double, integer, and entity handle
- Arbitrary data: stored as array of bytes

Other ITAPS Stuff

iGeom — Geometry Management

- Topological adjacency essentially identical to iMesh
- Interface for CAD data
  - Nearest point, projection, normal, curvature, etc
  - Physical and parametric coordinates
- Modest support for geometry manipulation

iRel — Relating Entities in Different Interfaces

- Classification of mesh entities / sets onto geometric entities
- Also usable for mesh-mesh or field-mesh relationships

ITAPS Mesh Interface Functionality

Core Mesh Functionality

Database Functions

- Load
- Save
- Geometric dimension
- Adjacency table

ITAPS Mesh Interface Functionality

Core Mesh Functionality

Database Functions

- Load
- Save
- Geometric dimension
- Adjacency table

Queries for Entire Entity Set

- Number of entities of given type or topology
- Entities of a given type or topology
- Coordinates of all vertices or for an array of vertices
- Adjacencies for all entities in set
- Vertex indices for all adjacent entities

ITAPS Mesh Interface Functionality

Entity and Block Access

Iterators

- Entity-by-entity or blocks of user-defined size
- Can iterate by entity type or topology
- Functionality: initialize, increment and return data, reset, destroy

ITAPS Mesh Interface Functionality

Entity and Block Access

Iterators

- Entity-by-entity or blocks of user-defined size
- Can iterate by entity type or topology
- Functionality: initialize, increment and return data, reset, destroy

Entity and Block Queries

- Single entity or array of entities
- Entity type and topology
- Vertex coordinates
- Adjacencies (first and second)

ITAPS Mesh Interface Functionality

Mesh Modification

Vertex Modification

- Vertex creation with coordinates
- Changing vertex coordinates
- No checks for valid location

ITAPS Mesh Interface Functionality

Mesh Modification

Vertex Modification

- Vertex creation with coordinates
- Changing vertex coordinates
- No checks for valid location

Topology Changes

- By deletion and creation, rather than modification
- Deletion top-down only
- Creation from equal dimension entities
- No checks for valid topology

ITAPS Mesh Interface Functionality

Entity Shape

Mesh Functions

- Set default shape for all entities
- Defined shape types: Lagrange, Bezier, cubic spline, analytic.

Entity Functions

- Set/retrieve high-order nodes
  - Simple for equal-order entities; more work for mixed-order
  - Parametric or physical coordinates
- Get entity on which a HO node lies
- Increase/decrease order

ITAPS Mesh Interface Functionality

Entity Set Interface

Set Operations

- Create, destroy, is ordered?

Set Membership

- Add / remove set as member
- Retrieve all member sets / number of member sets
- Check whether a set is contained in set

Entity Membership

- Add / remove (array of) entity as member(s)
- Queries described previously
- Check whether an entity is contained in set

ITAPS Mesh Interface Functionality

Set Relations and Boolean Operations

Heirarchical Set Relationships

- Create / destroy parent-child link
- Is child?
- Retrieve parents / children

Set Boolean Operations

- Intersection
- Union
- Subtraction

ITAPS Mesh Interface Functionality

Tags

Basic Tag Functionality

- Create / destroy
- Retrieve by name or handle
- Get all tags associated with an entity or set

ITAPS Mesh Interface Functionality

Tags

Basic Tag Functionality

- Create / destroy
- Retrieve by name or handle
- Get all tags associated with an entity or set

Setting and Retrieving Data

- Set / get double, integer, handle, and generic data
- Applied to entities, entity arrays, or entity sets

Language Interoperability for C and Fortran

- Basic API is a C API
- Modifications for Fortran compatibility
  - All functions are `void` functions
  - Added argument in C API for string length
  - Fortran code must accept call-by-value extension
- Auto-name mangling support

Language Interoperability for Python and Java Using SIDL/Babel

SIDL / Babel

- SIDL is an interface description language
- Babel compiles SIDL specs $\longrightarrow$ stubs and glue code
- Language interoperability via hidden common representation
- Supported languages: C, C++, F77, F90, Java, Python
- Downside: high overhead

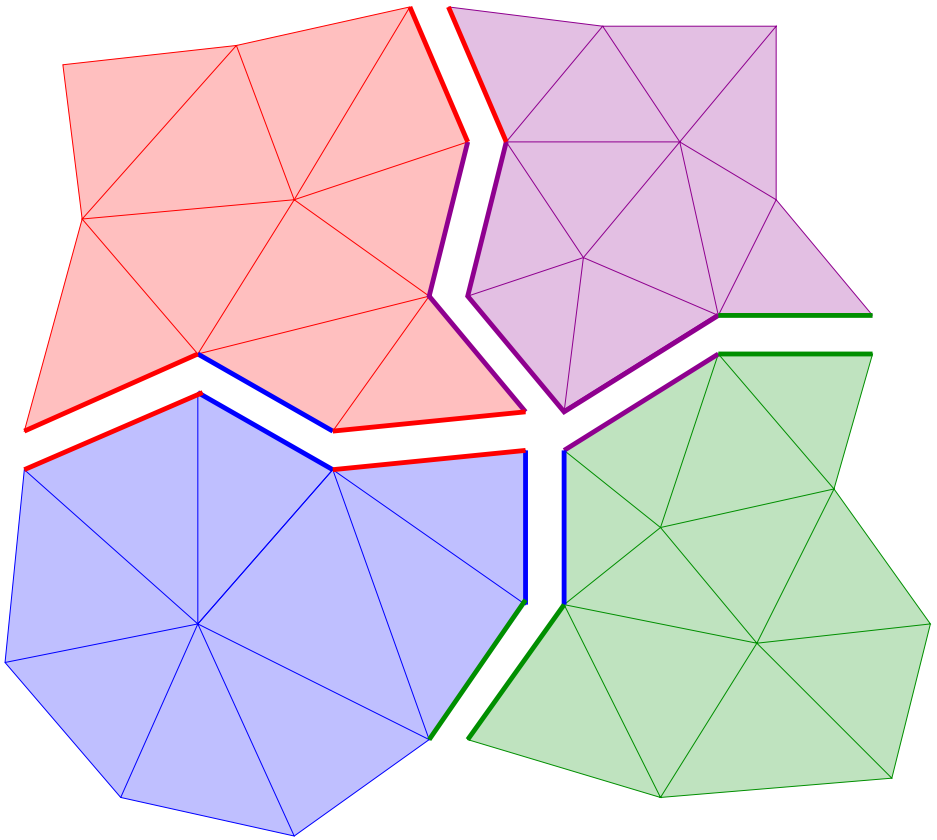
Usage

- Create ITAPS objects using factories
- Make ITAPS calls through Babel-generated client-language stubs
- Link to Babel runtime libraries

Parallel Data Model

Definitions

- A *Partition* is a covering of a mesh
- Subsets created by partitioning are called *Parts*
- Each *Process* operates on one or more parts



Parallel Functionality

Partitions and Parts

Partition Creation and Management

- Create/destroy partitions and parts
- Mapping between parts and processes
- Collective parallel queries for total numbers of entities

Parallel Functionality

Partitions and Parts

Partition Creation and Management

- Create/destroy partitions and parts
- Mapping between parts and processes
- Collective parallel queries for total numbers of entities

Part-level Queries

- Queries that work in serial also work on a part in parallel
- Identify part neighbors
- Identify entities on part bdrys

Parallel Functionality

Support for Migration and Adaptivity

Load Balancing and Re-partitioning

- Large-scale mesh data "push"
- Non-blocking send, blocking receive
- Intended to be a collective call

Parallel Functionality

Support for Migration and Adaptivity

Load Balancing and Re-partitioning

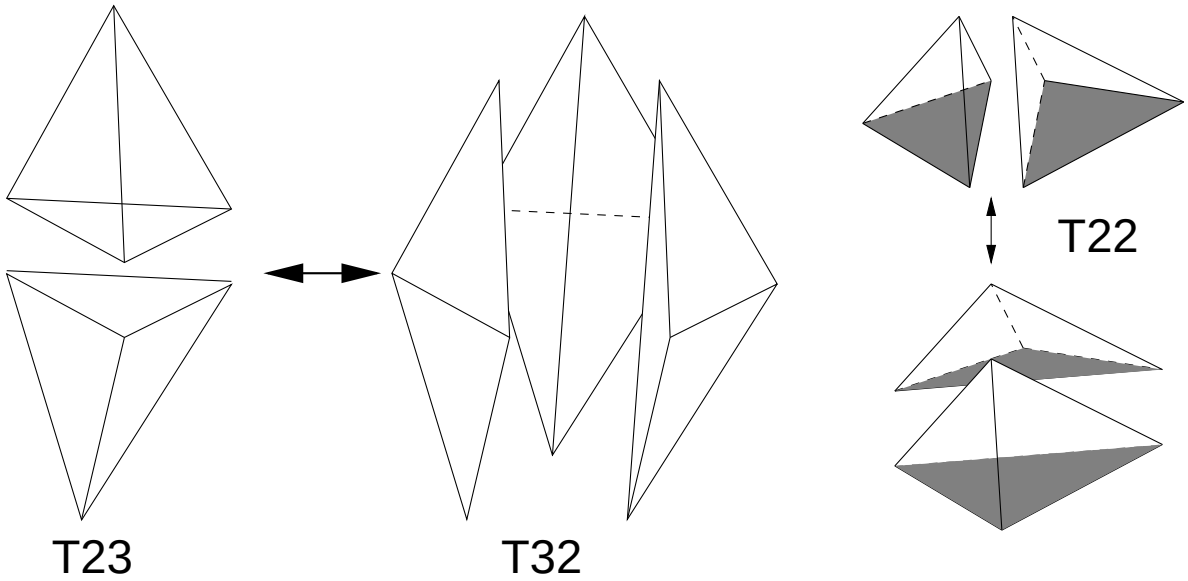
- Large-scale mesh data "push"
- Non-blocking send, blocking receive
- Intended to be a collective call

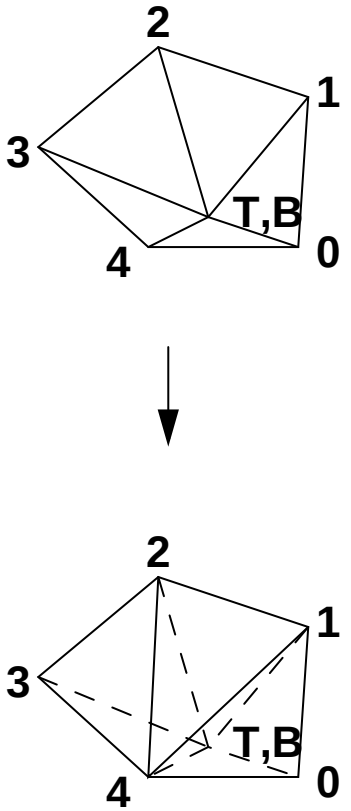
Small-Scale Migration

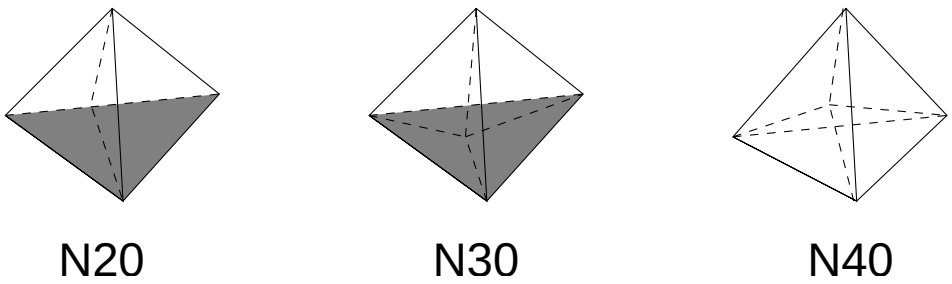
- Non-blocking, point-to-point requests
  - Change of ownership, vertex re-location, matching up new entities on part bdry, etc
- May generate multiple rounds of point-to-point communication
- Processes must poll for and handle messages

ITAPS Swapping Service

Swapping Operations







ITAPS Swapping Service

Features

Swapping Features

- Face- and edge-swapping
- Several standard swapping criteria built in, plus user-defined

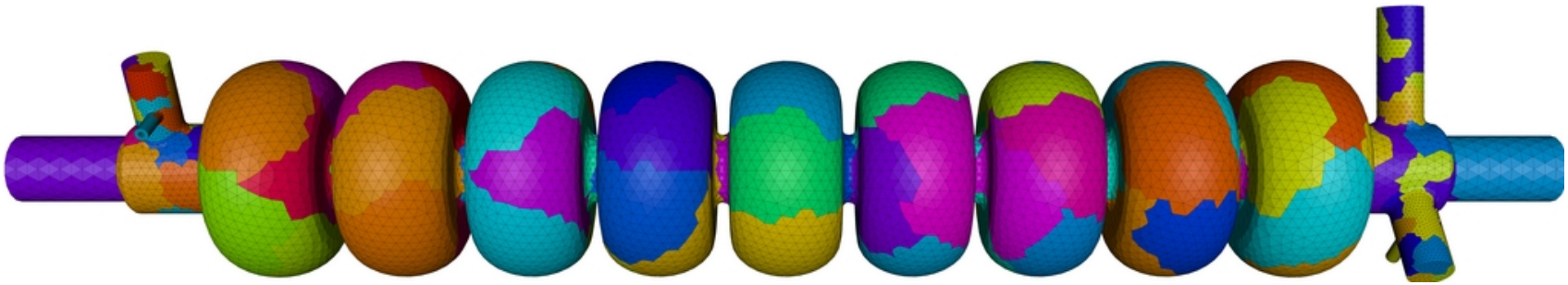
ITAPS Usage

- Requires approximately a dozen iMesh functions
  - Adjacencies, iterators, create/delete entities
  - Uses block retrieval where appropriate for efficiency
  - Optional use of sets and tags to indicate what to swap
- Also uses a handful of other ITAPS functions

ITAPS Application Example

SLAC Accelerator Design Optimization

- Parallel mesh generation and adaptation (uses iMesh on each part)
- ITAPS-based geometry interface used for curved bdry data
- Shape optimization loop is an ITAPS service



iMesh Implementations, Services, and Applications

Status and Plans

- Currently four full implementations of (serial) iMesh
- Half a dozen working services (mesh improvement, front tracking, optimization, adaptation)
- Parallel interface roll-out late spring; implementations by late summer / fall (before Supercomputing)
- Work with various DOE SciDAC applications is on-going

Support

- US Department of Energy under the Scientific Discovery through Advanced Computation (SciDAC) program.
- The Canadian Natural Sciences and Engineering Research Council under a Special Research Opportunities grant.

For More Information:

`http://www.itaps-scidac.org` or email `cfog@mech.ubc.ca`.