7th Annual Workshop on Charm++ and its Applications

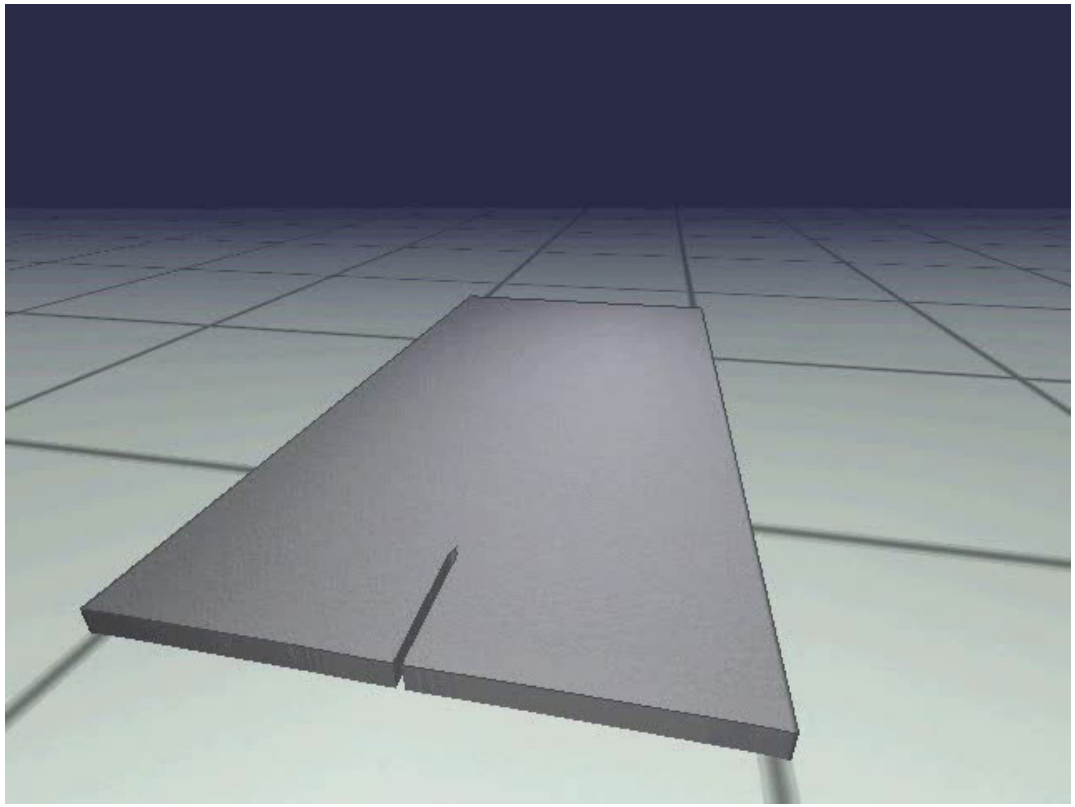
ParTopS: Compact Topological Framework for Parallel Fragmentation Simulations

Rodrigo Espinha[1]
Waldemar Celes[1]
Noemi Rodriguez[1]
Glaucio H. Paulino[2]

1. Computer Science Dept., Pontifical Catholic University of Rio de Janeiro, Brazil
2. Dept. of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign

Motivation

- Fragmentation simulations using extrinsic cohesive models
  - Evolutive problems in space and time
  - Cohesive elements inserted dynamically
  - Highly refined mesh at crack tip region



ParTopS[1]

- Parallel framework for finite element meshes
  - Distributed mesh representation
    - Extension of the TopS[2] topological data structure
  - Parallel algorithm for inserting cohesive elements
    - Extension of the serial algorithm by Paulino et al.[3]

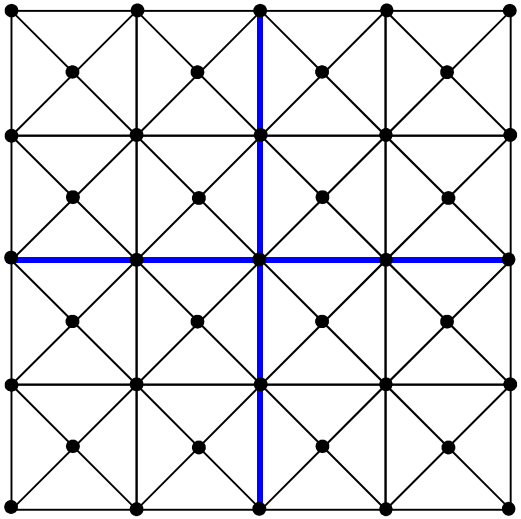
1. Espinha R, Celes W, Rodriguez N, Paulino GH (2009) ParTopS: Compact Topological Framework for Parallel Fragmentation Simulations. *Submitted to Engineering with Computers*

2. Celes W, Paulino GH, Espinha R (2005) A compact adjacency-based topological data structure for finite element mesh representation. Int J Numer Methods Eng 64(11):1529–1565

3. Paulino GH, Celes W, Espinha R, Zhang Z (2008) A general topology-based framework for adaptive insertion of cohesive elements in finite element meshes. Engineering with Computers 24(1):59-78

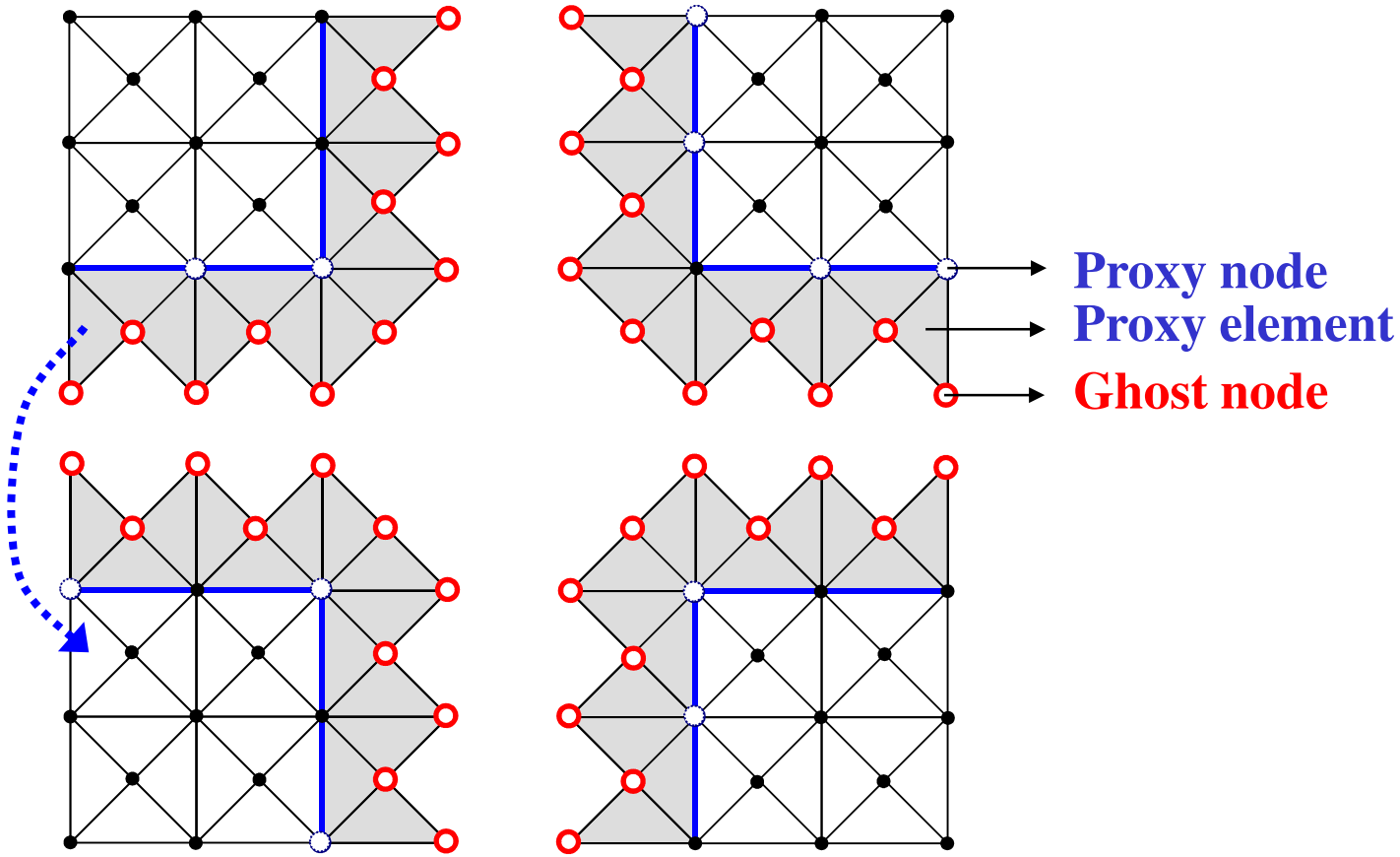
Distributed mesh representation

- Sample mesh



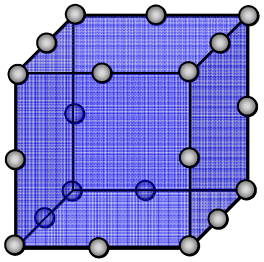
Distributed mesh representation

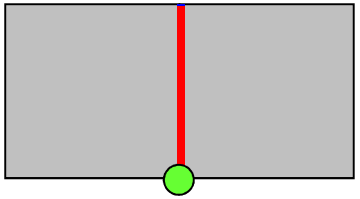
Communication layer

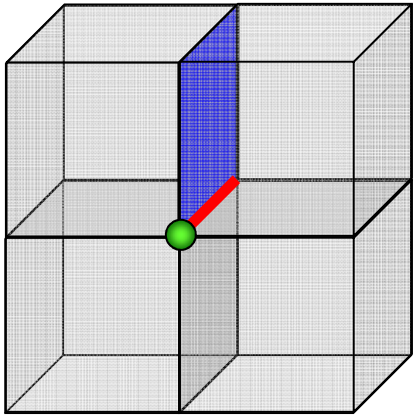


Topological entities

- Element
- Node
- Facet
  - Interface between elements
- Edge
- Vertex

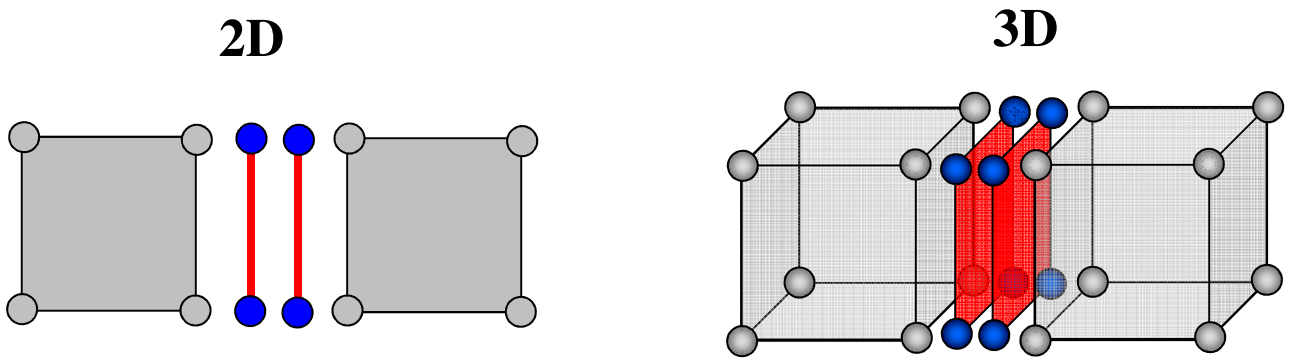






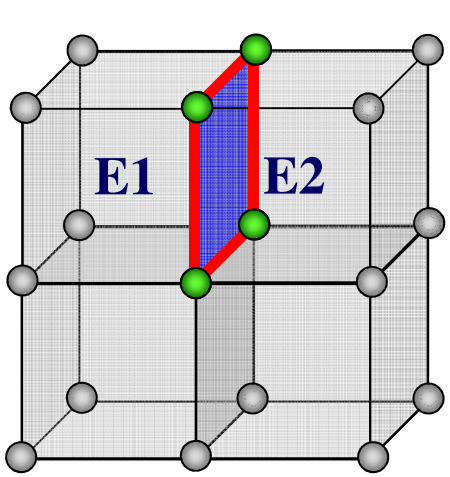
Cohesive elements

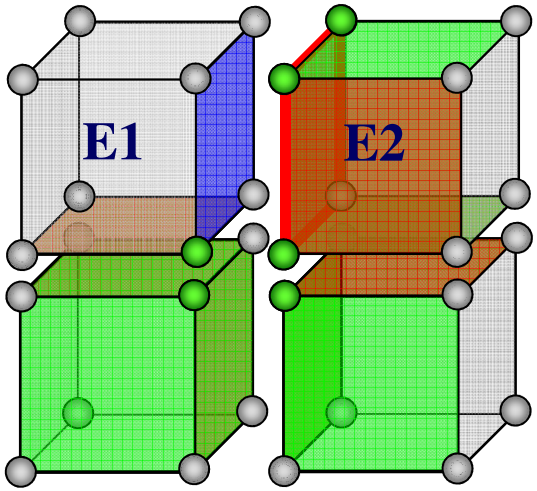
- True extrinsic elements
  - Inserted “on the fly”, where needed and when needed
  - No element activation or springs
- Two-facet elements
- Inserted between two adjacent bulk elements



Serial insertion of cohesive elements

- Insert cohesive element at a facet shared by E1 and E2
  1. Create cohesive element at facet
  2. Traverse non-cohesive elements adjacent to edges of E2
     - If E1 is not visited, duplicate edge and mid-nodes (if any)
  3. Traverse non-cohesive elements adjacent to vertices of E2
     - If E1 is not visited, duplicated vertex

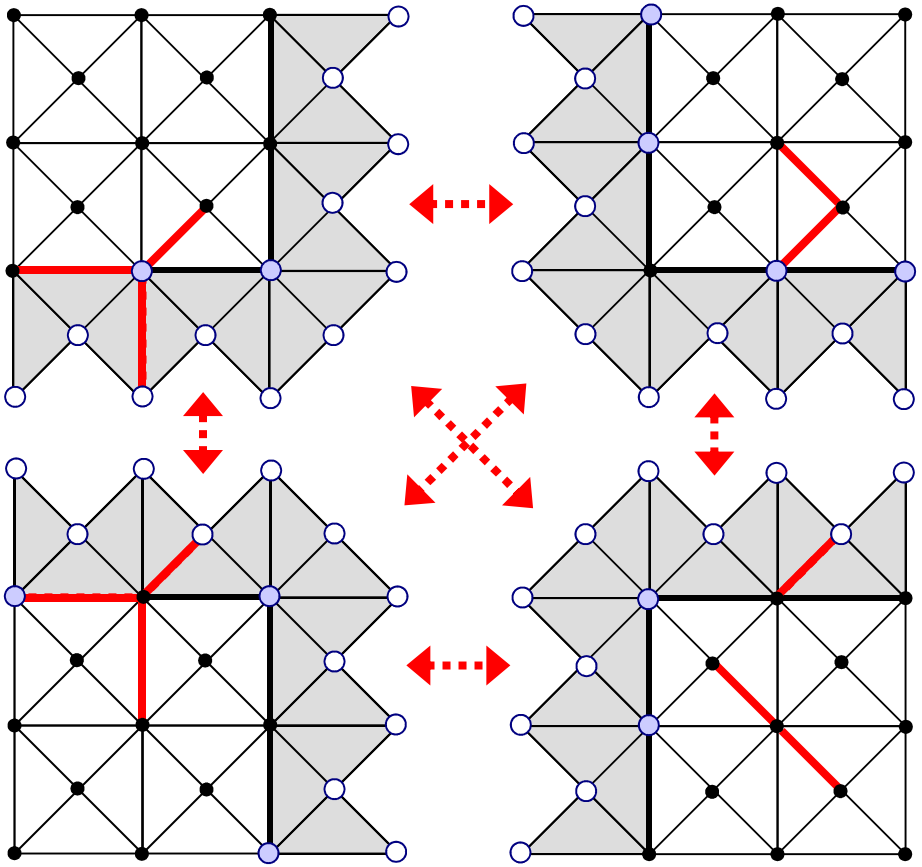




Parallel insertion of cohesive elements

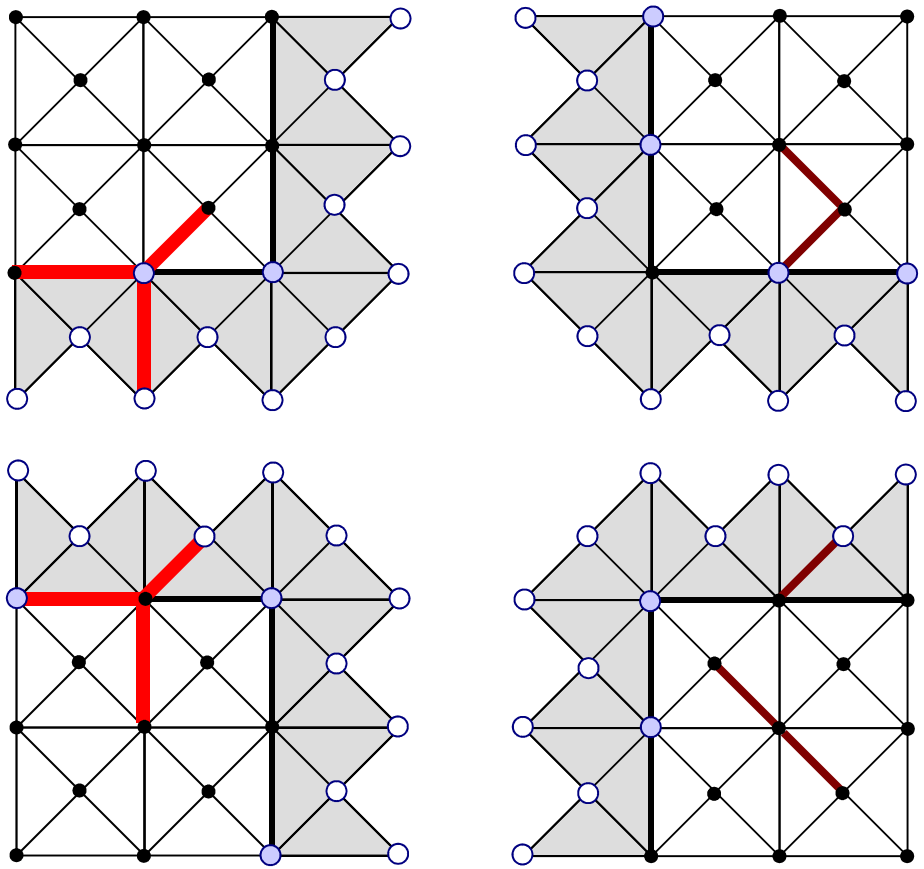
- At each simulation step
  - Analysis application identifies fractured facets
  - Insert cohesive elements
    1. Insert elements at local and proxy facets
    2. Update new proxy entities
    3. Update affected ghost entities

Identification of fractured facets

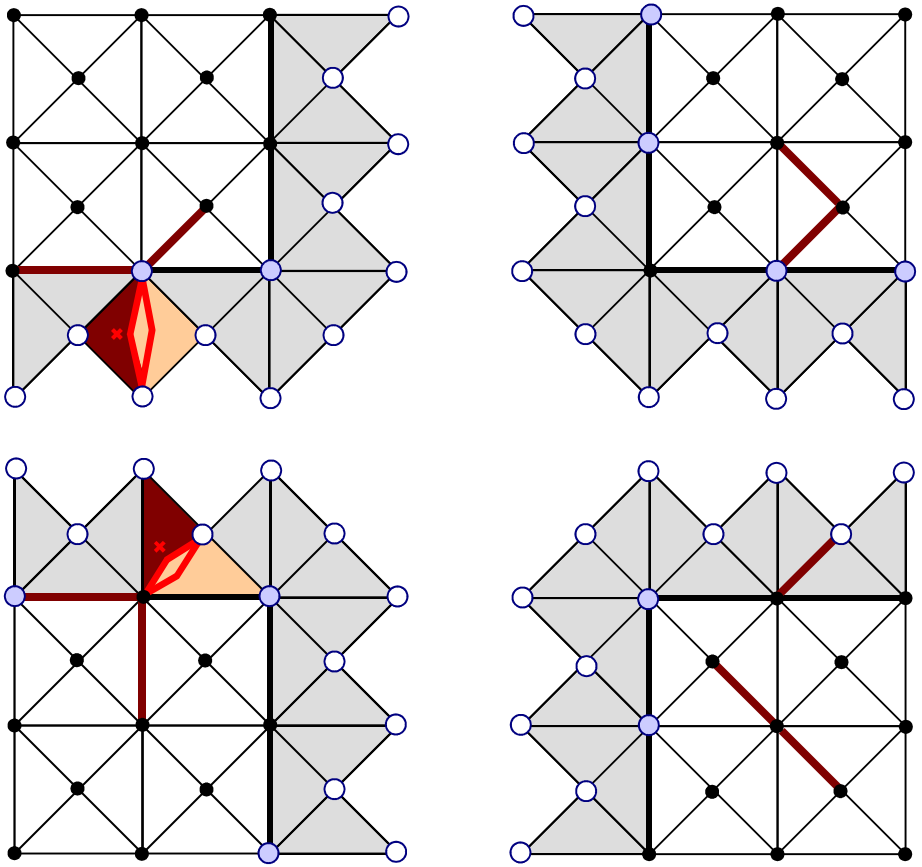


1. Insert elements at local and proxy facets

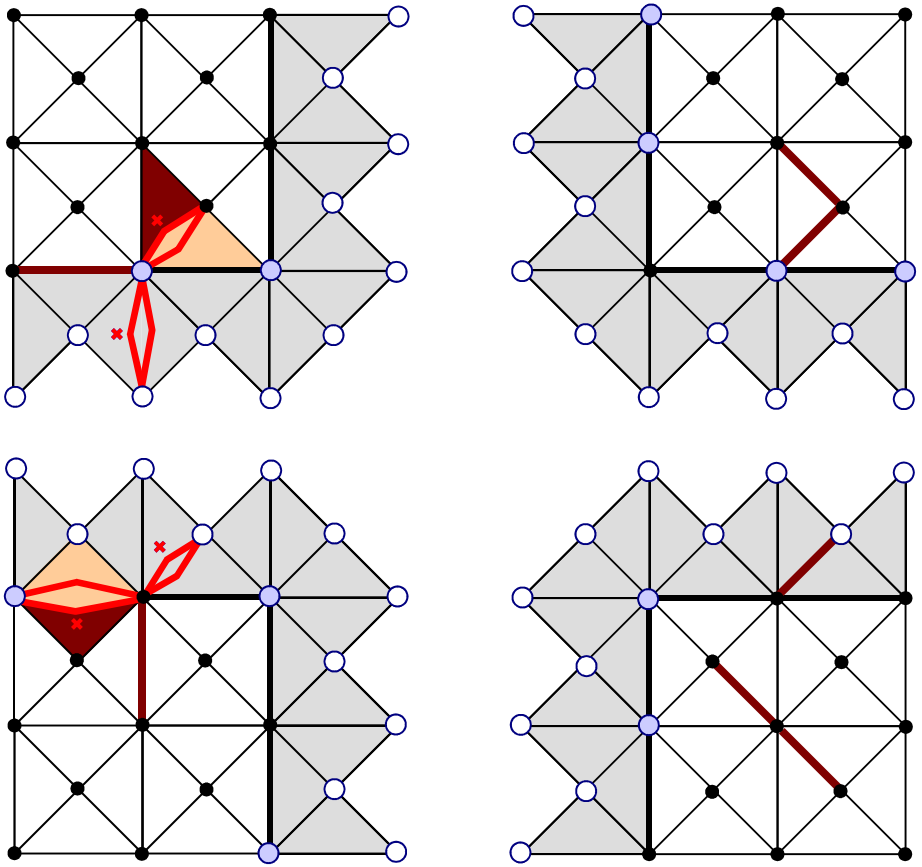
- Serial algorithm with additional constraints
  - Ghost nodes are not duplicated at this moment
    - Dependence on remote information
  - All the copies of a new element or node must be owned by the same partition



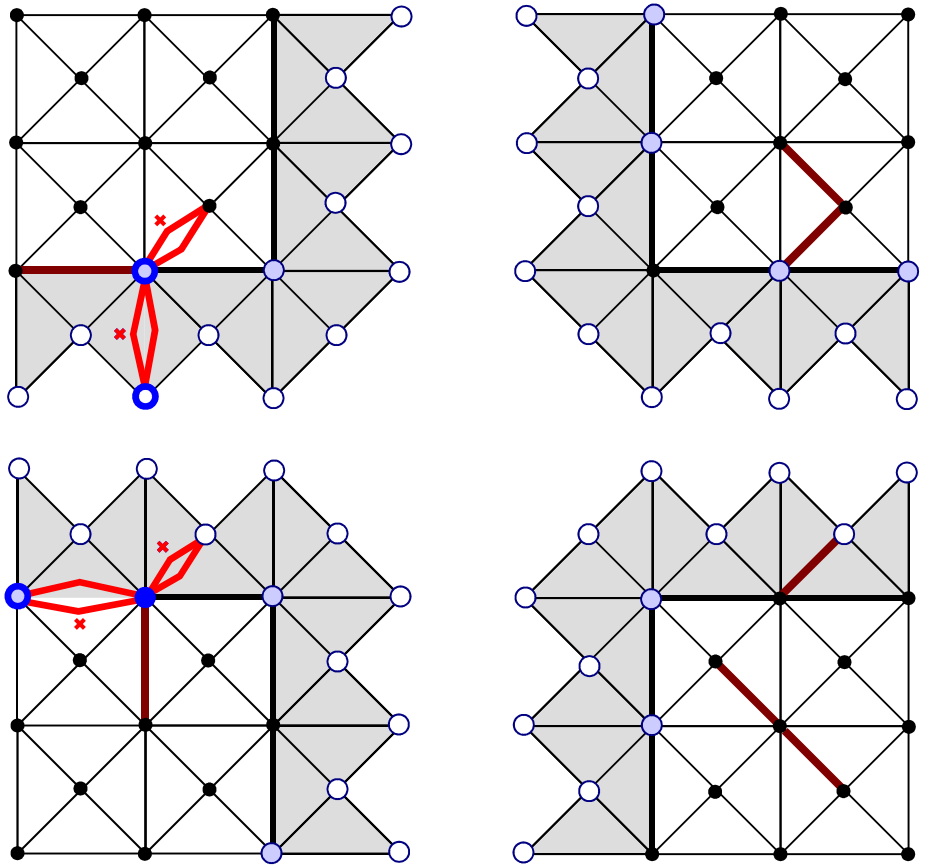
1. Insert elements at local and proxy facets



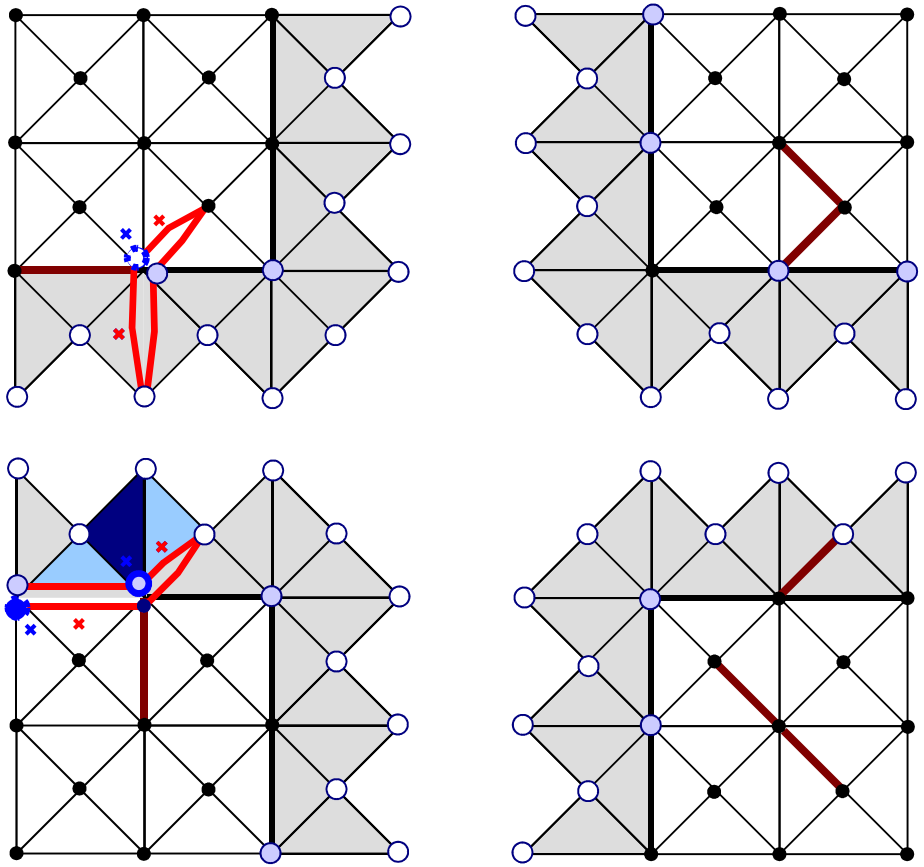
1. Insert elements at local and proxy facets



1. Insert elements at local and proxy facets

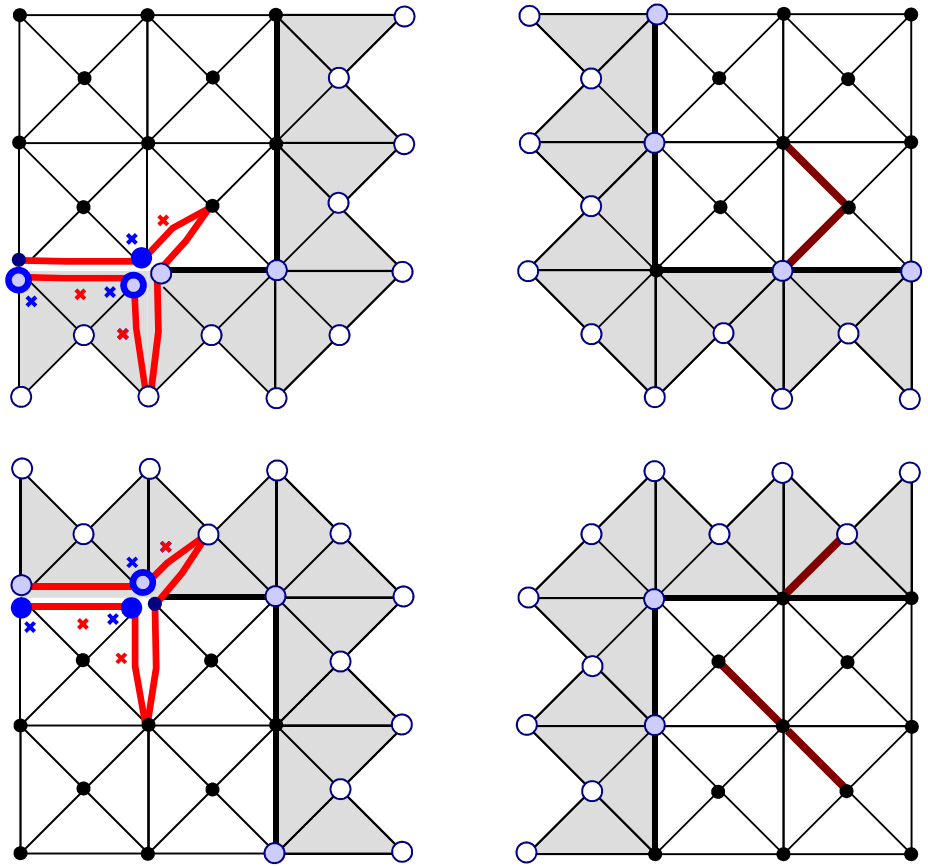


1. Insert elements at local and proxy facets

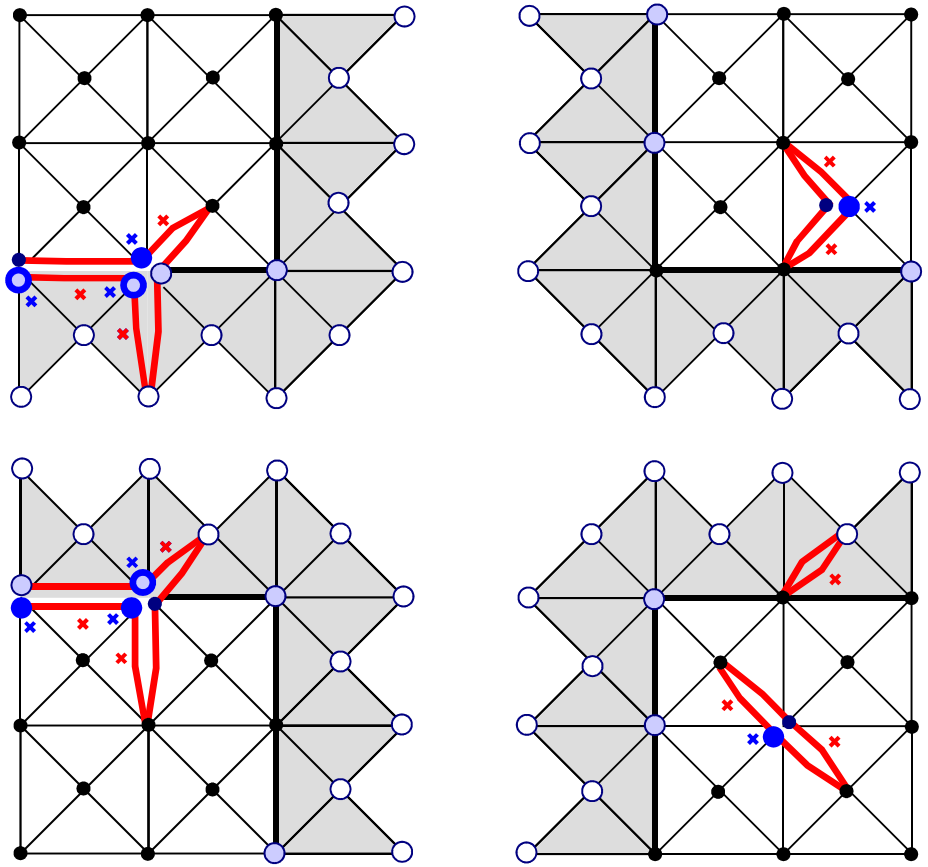


1. Insert elements at local and proxy facets

- Uniform criterion for selecting representative elements
  - E.g. the adjacent element with smallest (partition_id, element_id)
- Consistent topological results in both partitions

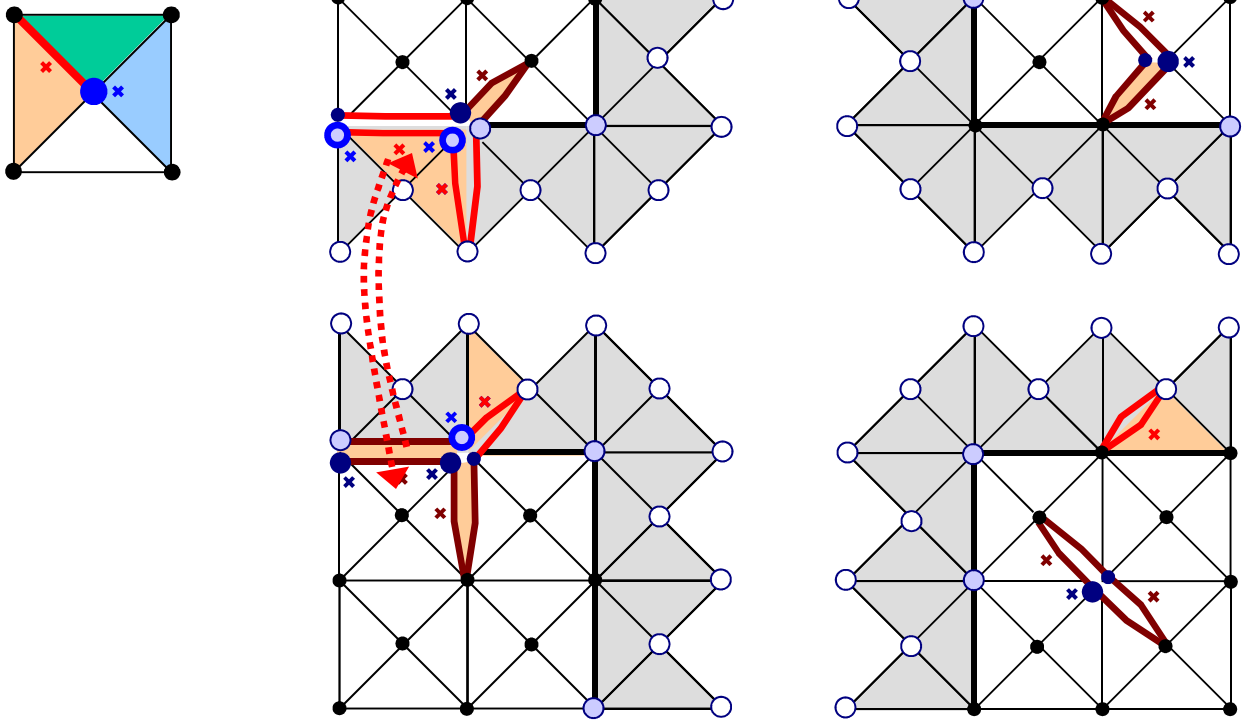


1. Insert elements at local and proxy facets

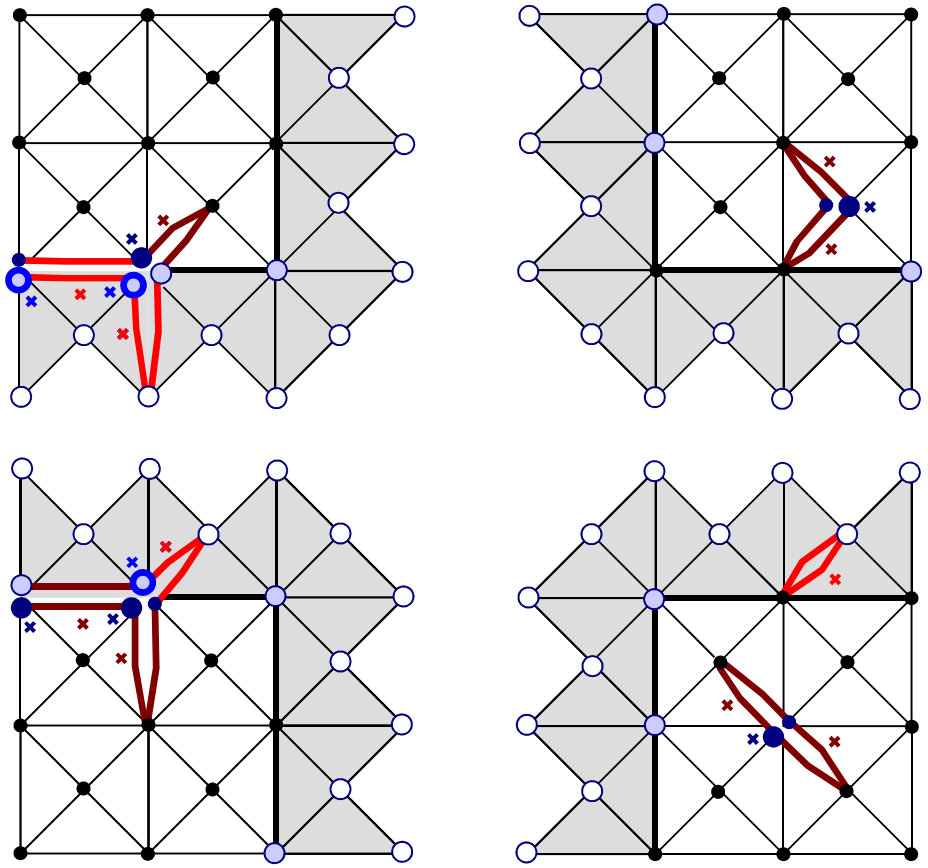


2. Update new proxy entities

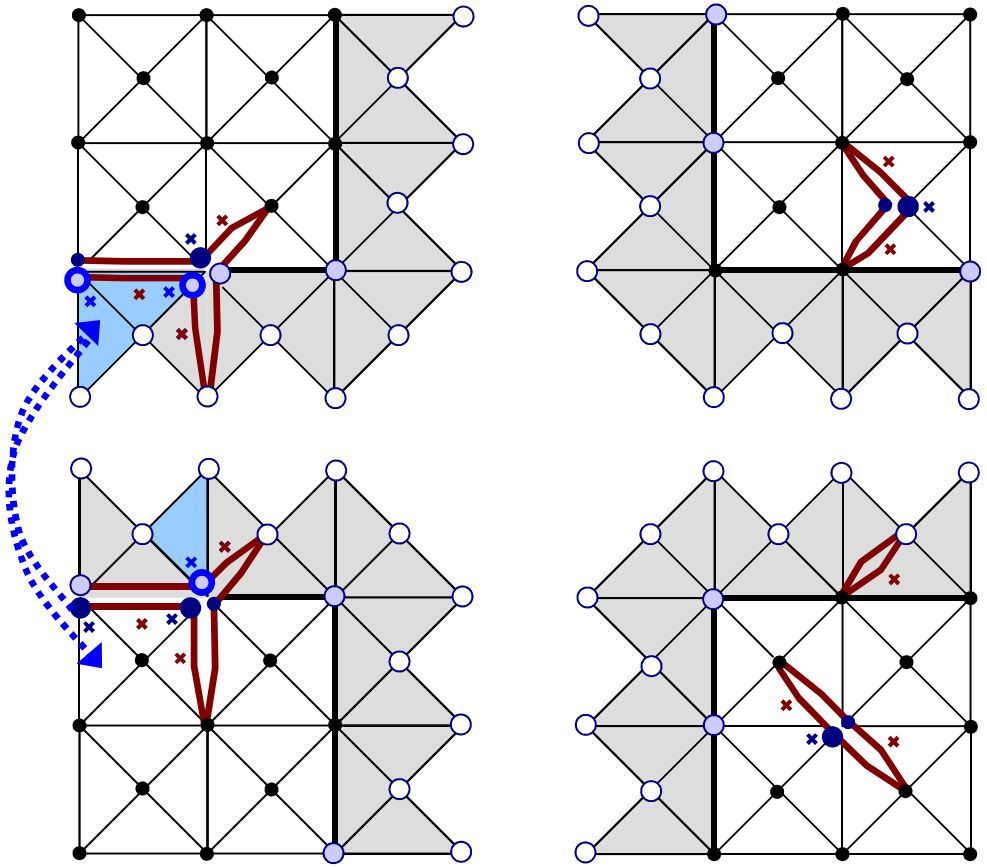
- Create references from the new proxy elements and nodes to the corresponding real entities



2. Update new proxy entities

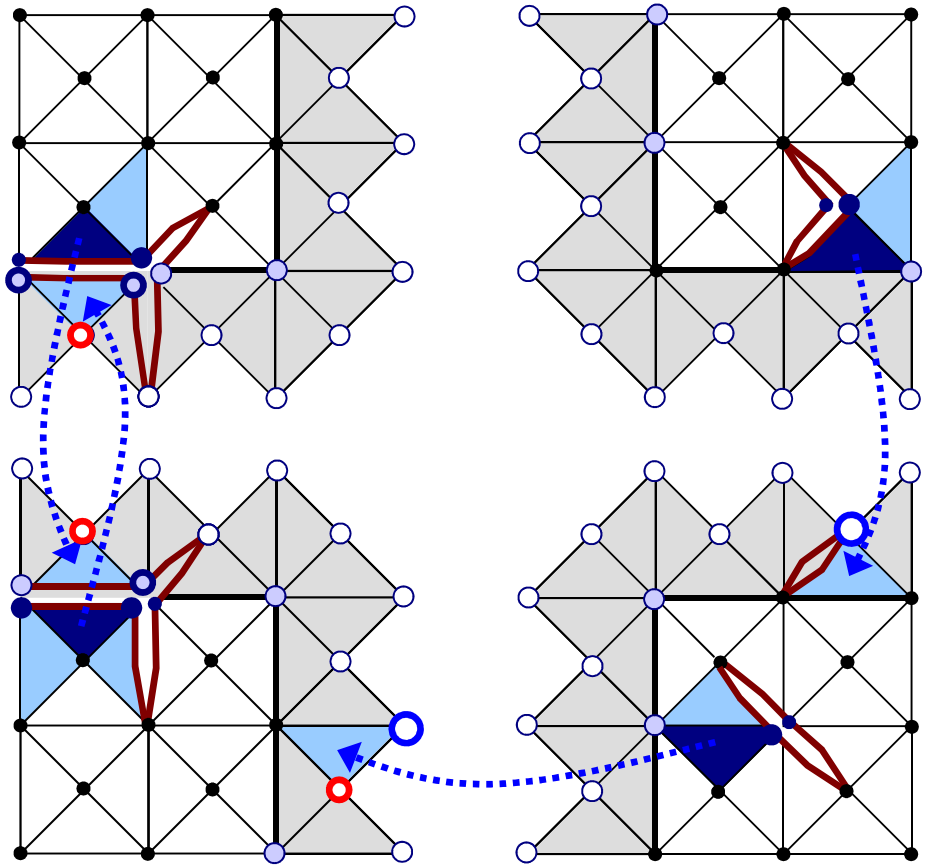


2. Update new proxy entities

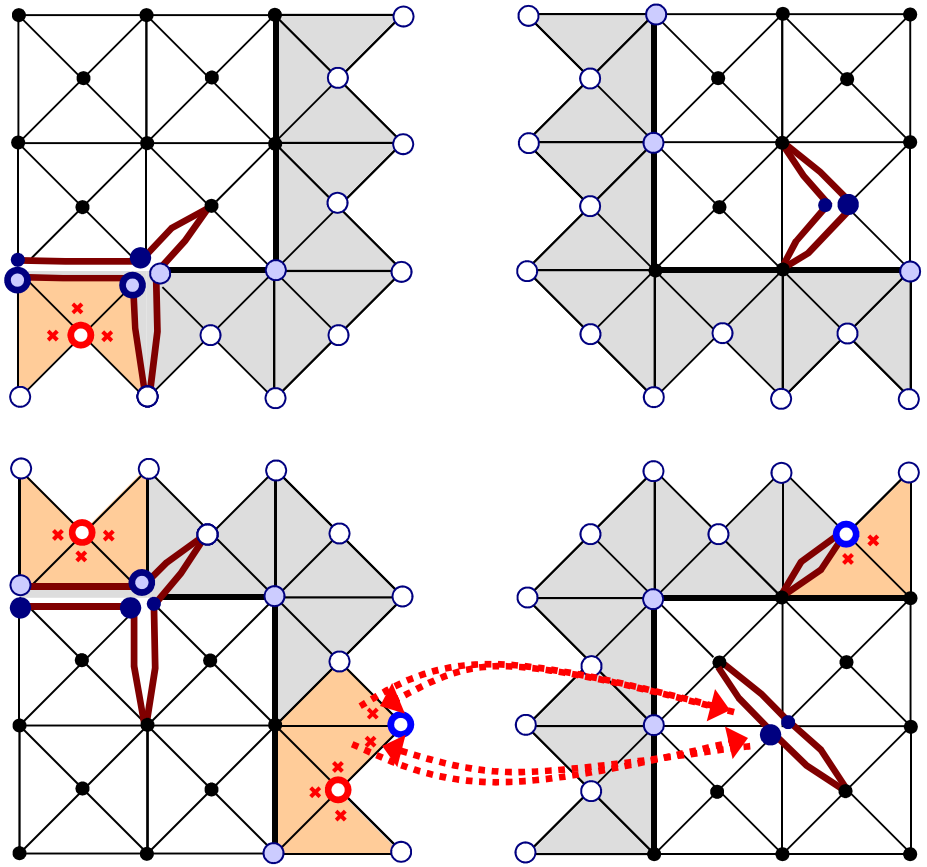


3. Update affected ghost entities

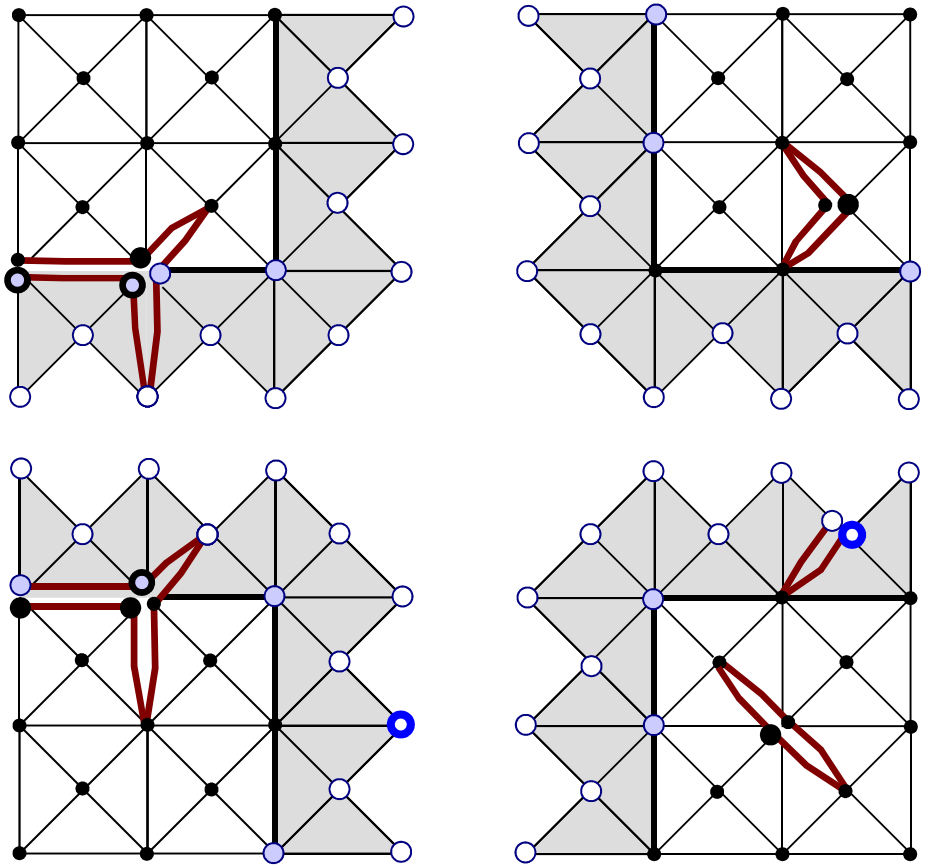
- Replace ghost nodes affected by remote cohesive elements
  - "Per-element" approach
    - Partitions with elements adjacent to duplicated nodes notify others



3. Update affected ghost entities

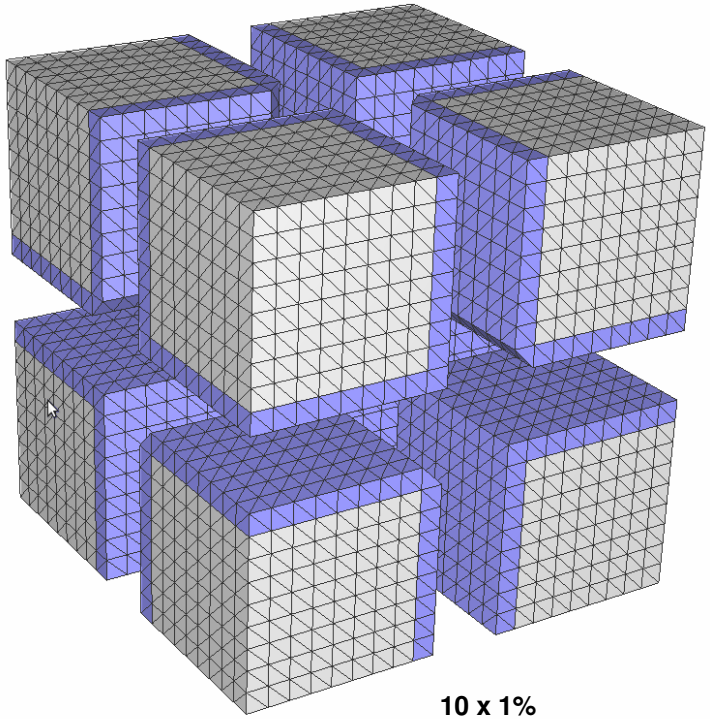


Resulting mesh

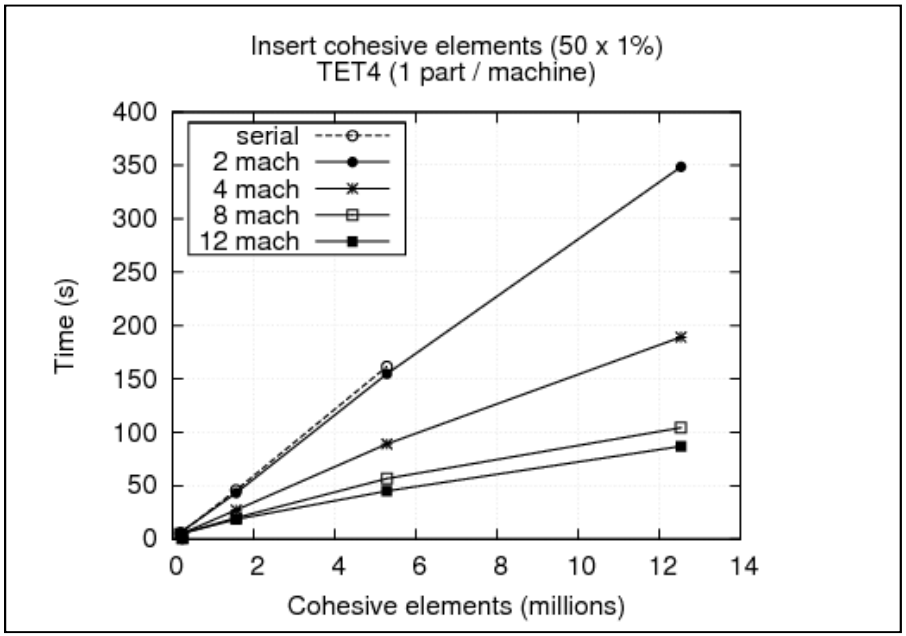


Verification

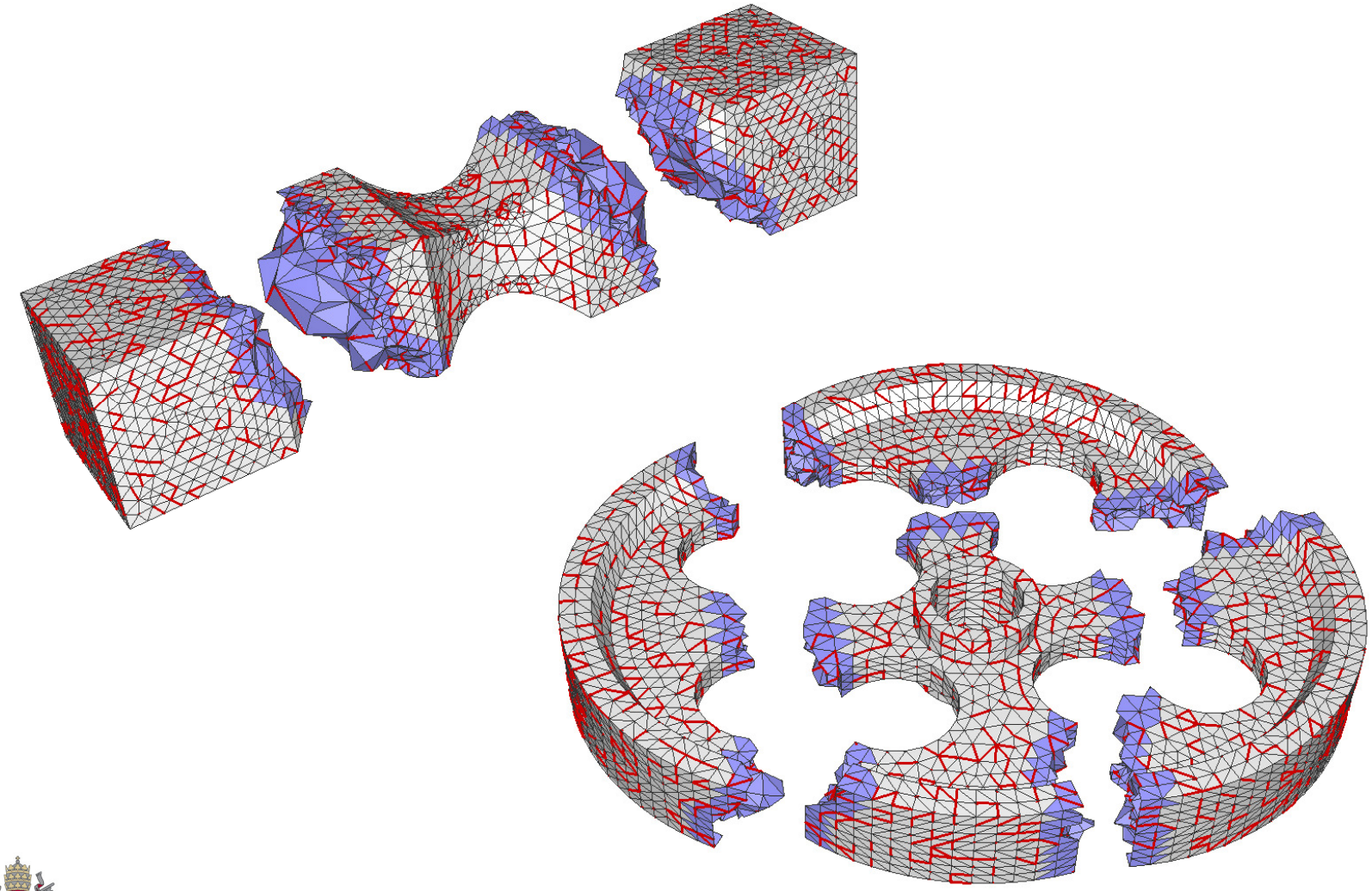
- Cluster of 12 machines
  - Intel(R) Pentium(R) D processor 3.40 GHz (dual core) with 2GB of RAM, Gigabit Ethernet
- Cohesive elements randomly inserted at 1% of internal facets x 50 steps
- Meshes with different discretizations and types of elements (T3, T6, Tet4, Tet10)



10 x 1%



Results



Summary

- ParTopS: parallel topological framework
  - Dynamic insertion of cohesive elements
    - True extrinsic cohesive elements
      - Inserted “on the fly”, where needed and when needed
    - Generic branching patterns are supported
    - General 2D or 3D meshes
    - Executed on a limited number of machines
      - However, linear scaling is expected

ParTopS
Implementation using Charm++

- Why Charm++?
  - Potential for optimization
  - Integrated load balancers
  - Set of available tools
- Implementation (so far)
  - Each mesh partition as a virtual processor (Chare)
  - Asynchronous method calls
  - SDAG used to manage control flow of the three phases of the algorithm
    - Cohesive elements created asynchronously on partitions' boundaries
    - Bulk updates of modified data
- Implementation (objectives)
  - Explore asynchronous communication in mesh related algorithms
  - Explore load balancing in parallel fragmentation applications

Future work

- Execute on a large number of processors
- Other parallel adaptive operators
  - E.g. refinement & coarsening
- Mechanical analysis computer code

Thank you!