Parallelization Of The Spacetime Discontinuous Galerkin Method Using The Charm++ FEM Framework (ParFUM)

Mark Hills, Hari Govind, Sayantan Chakravorty,
Terry Wilmarth, L.V. Kale, Robert Haber

presented by:

Isaac Dooley

University Illinois Urbana-Champaign

Overview

- My background in parallel programming
- How the Spacetime Discontinuous Galerkin Method utilizes unstructured meshes.
- Requirements to parallelize SDG
- Existing functionality in ParFUM which satisfies some SDG requirements
- New functionality which has been added to ParFUM to support the rest of the SDG requirements

Parallel Programming Lab

- Our focus is parallel programming, especially in frameworks and dynamic or adaptive applications
- We are not Computational Geometers, nor Mathematicians.
- We try to build general purpose reusable high performance frameworks
- Charm++ and AMPI
- Focus on Migratable Objects and Virtualization
- Multiple Platforms (Clusters, SMPs, BlueGene/L)

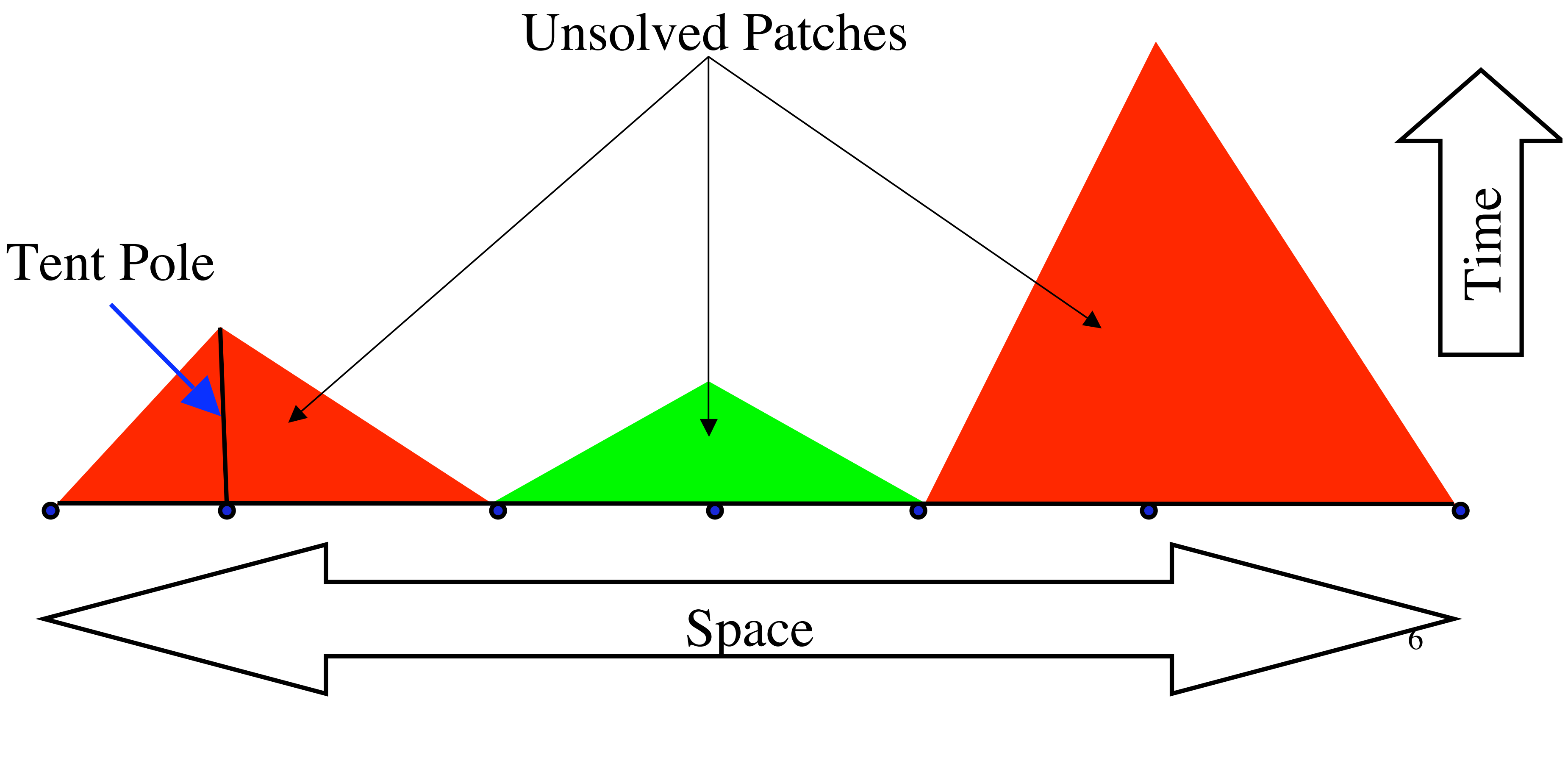
Spacetime Discontinuous Galerkin

- Collaboration with:
  - Bob Haber, Jeff Erickson, Mike Garland, …
  - NSF funded center
- SDG Motivation: Spatial adaptivity is needed in structural dynamics applications. Why shouldn't we also adapt in the time dimension?

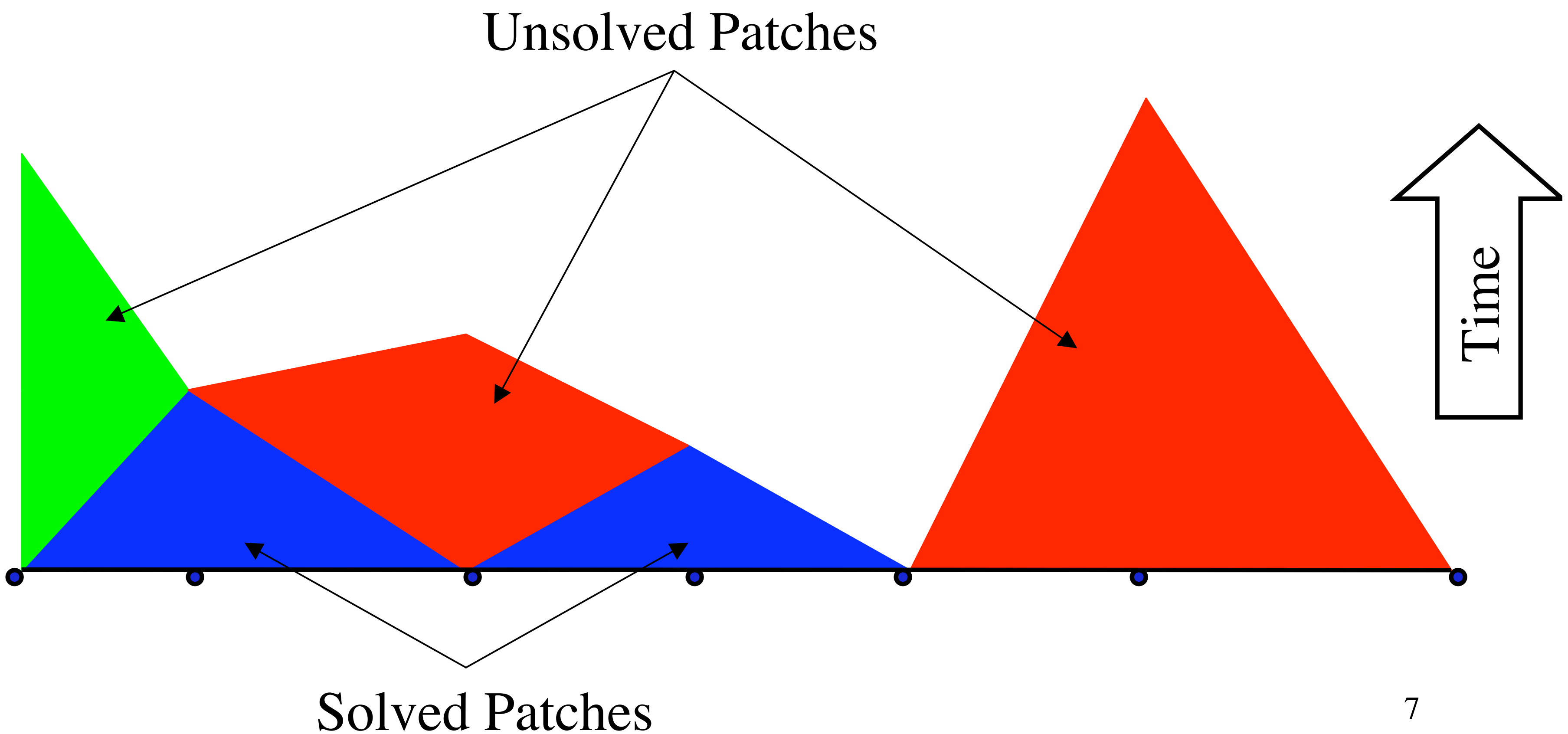
Spacetime Discontinuous Galerkin

- Mesh generation is an advancing front algorithm called Tent Pitcher.
- Adds a set of new elements called patches to the mesh, then solves them, thus advancing the front.
- Each patch depends only on inflow elements.

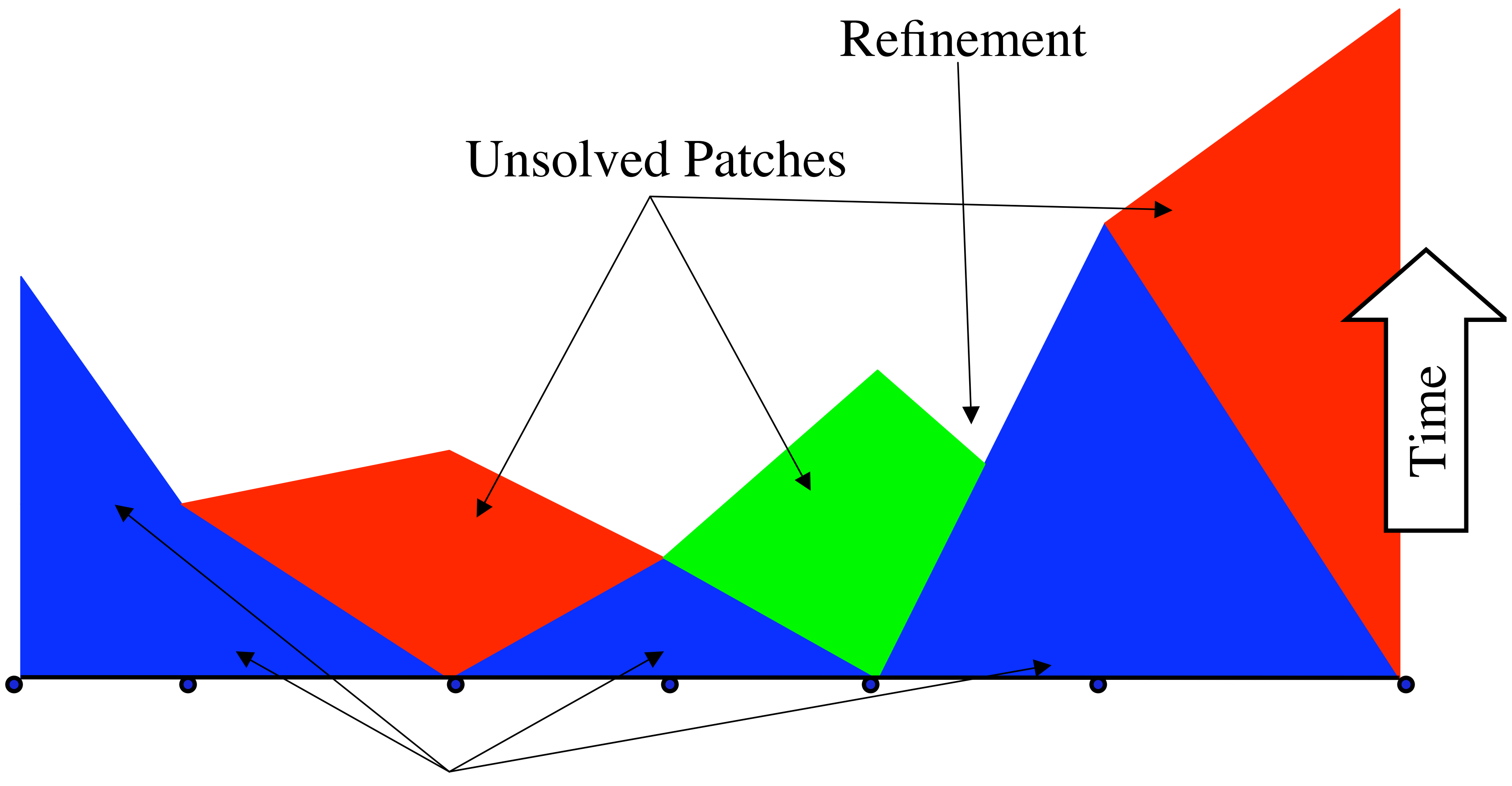
1-d Mesh Generation



1-d Mesh Generation



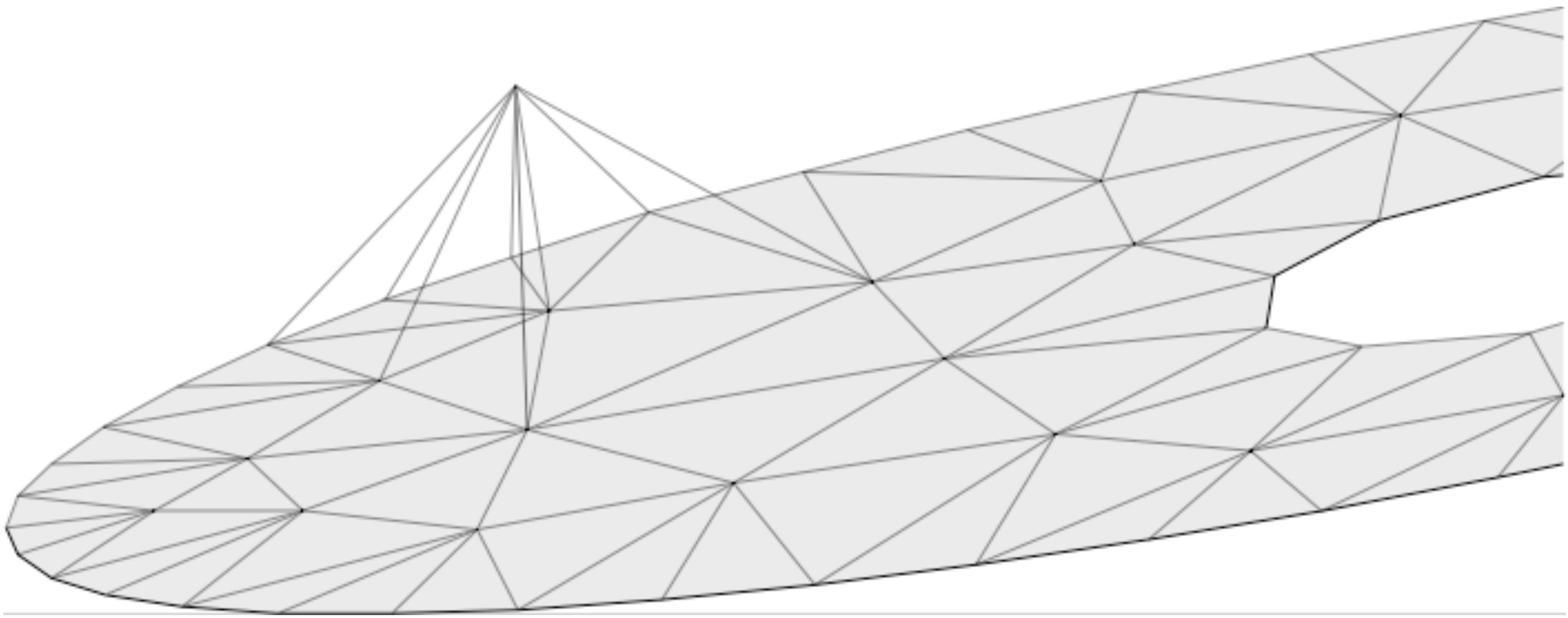
1-d Mesh Generation



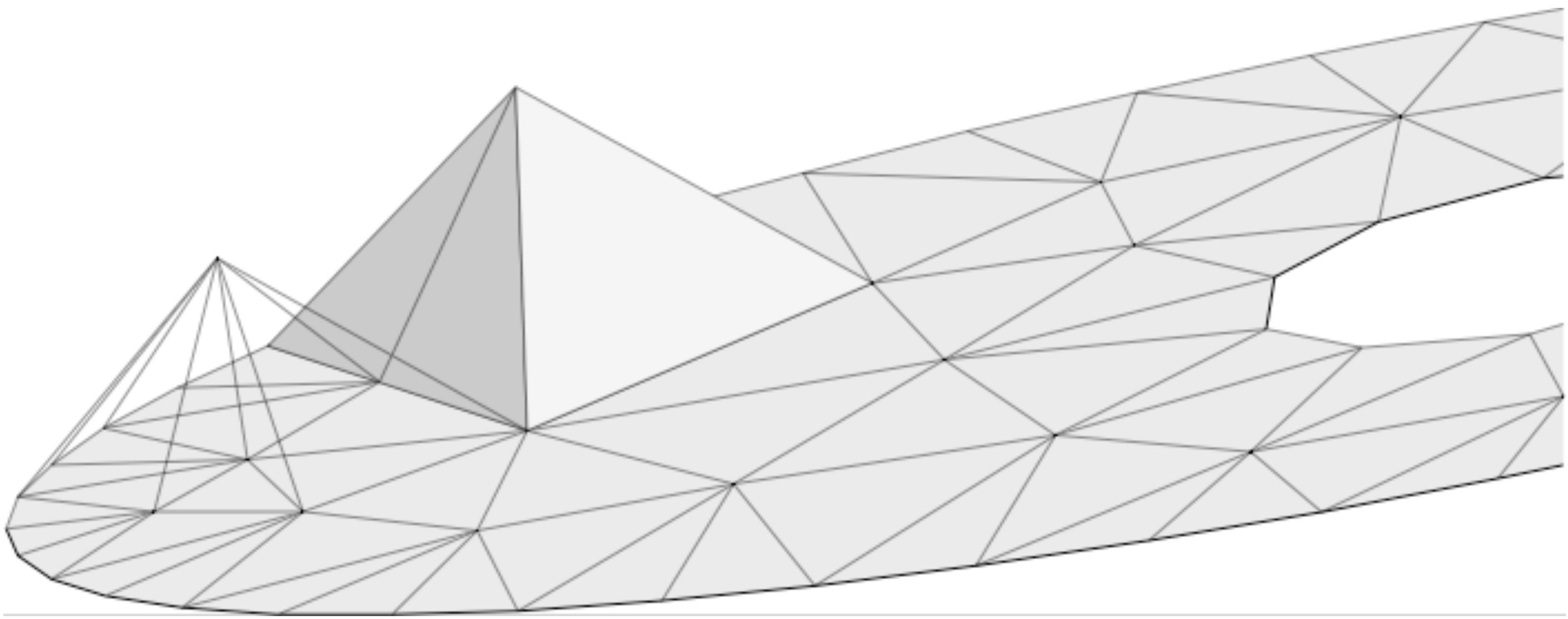
Adaptive SDG

- Method described in
  - Abedi, Zhou, et. al.; *Spacetime meshing with adaptive refinement and coarsening; 2004*
- Tent poles are not just pitched above existing space nodes
- Entire space-time mesh or frontier is built as a mesh. Non-adaptive SDG can store patches as attributes of nodes in original mesh.

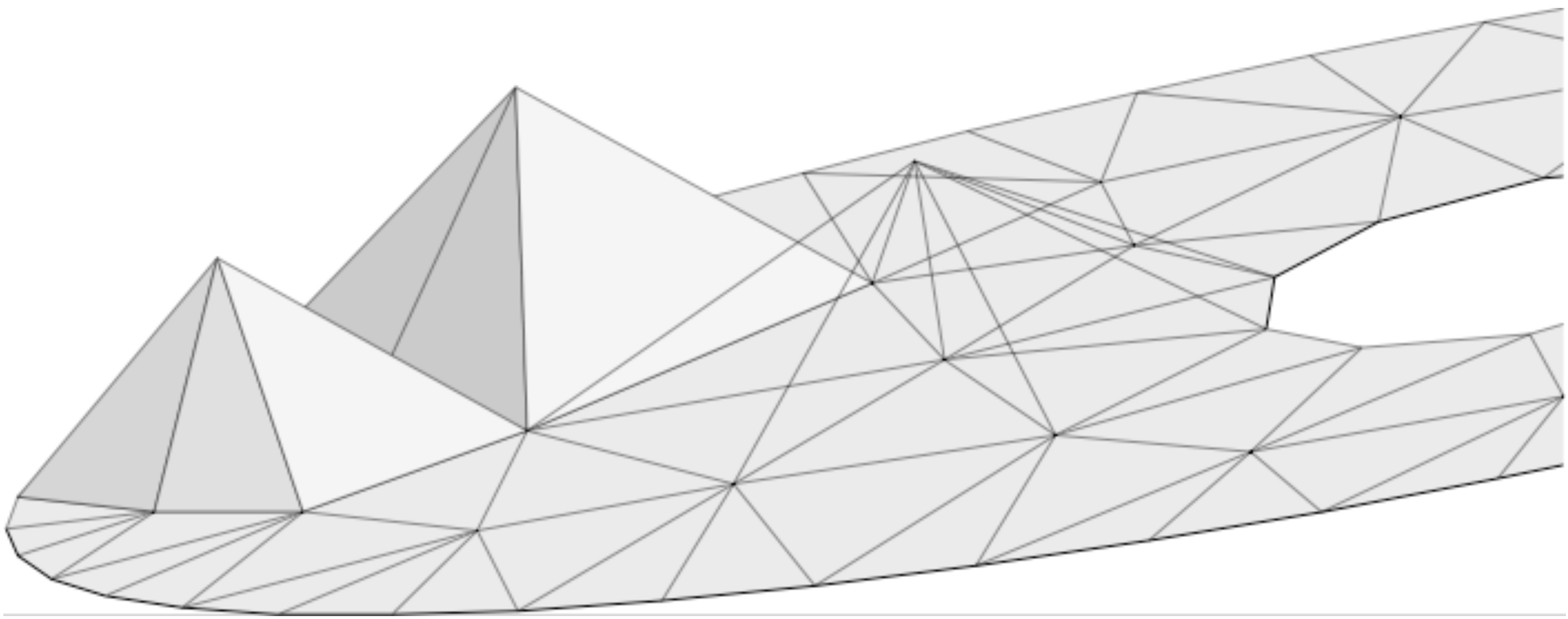
2-d Adaptive Mesh Generation



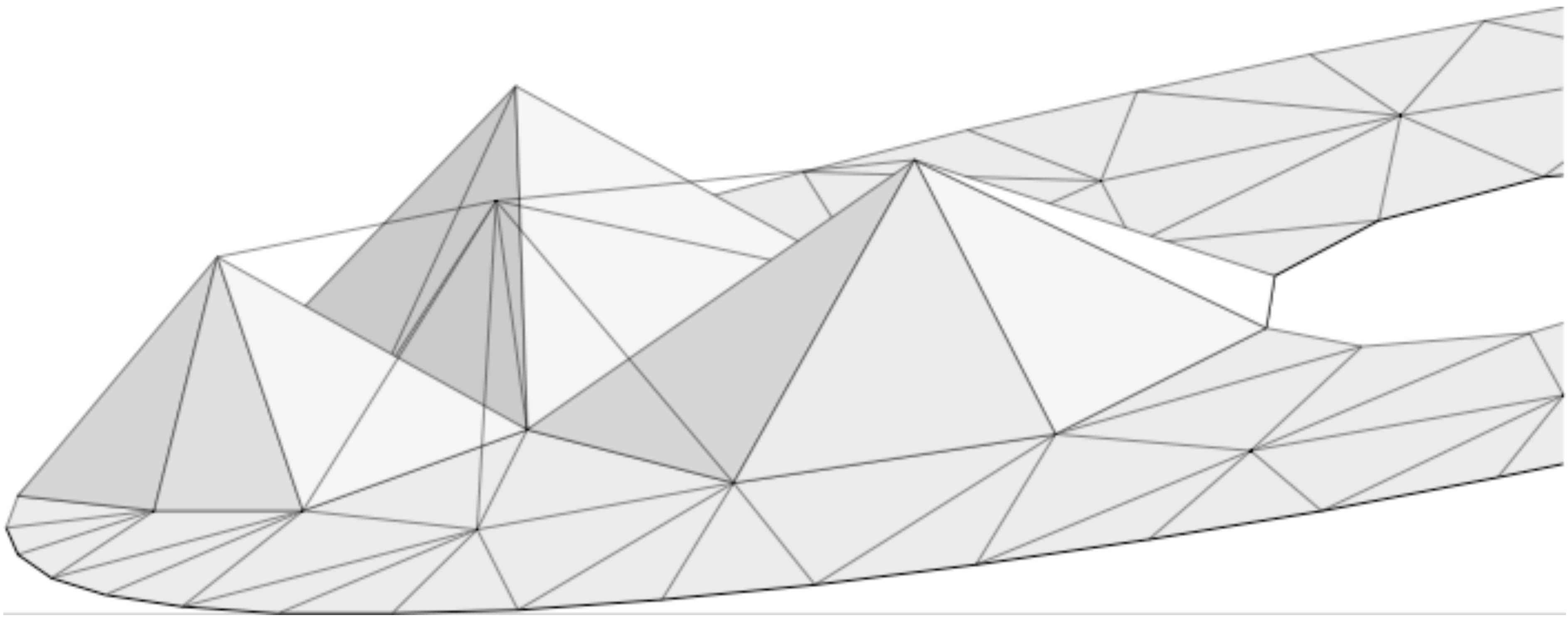
2-d Adaptive Mesh Generation



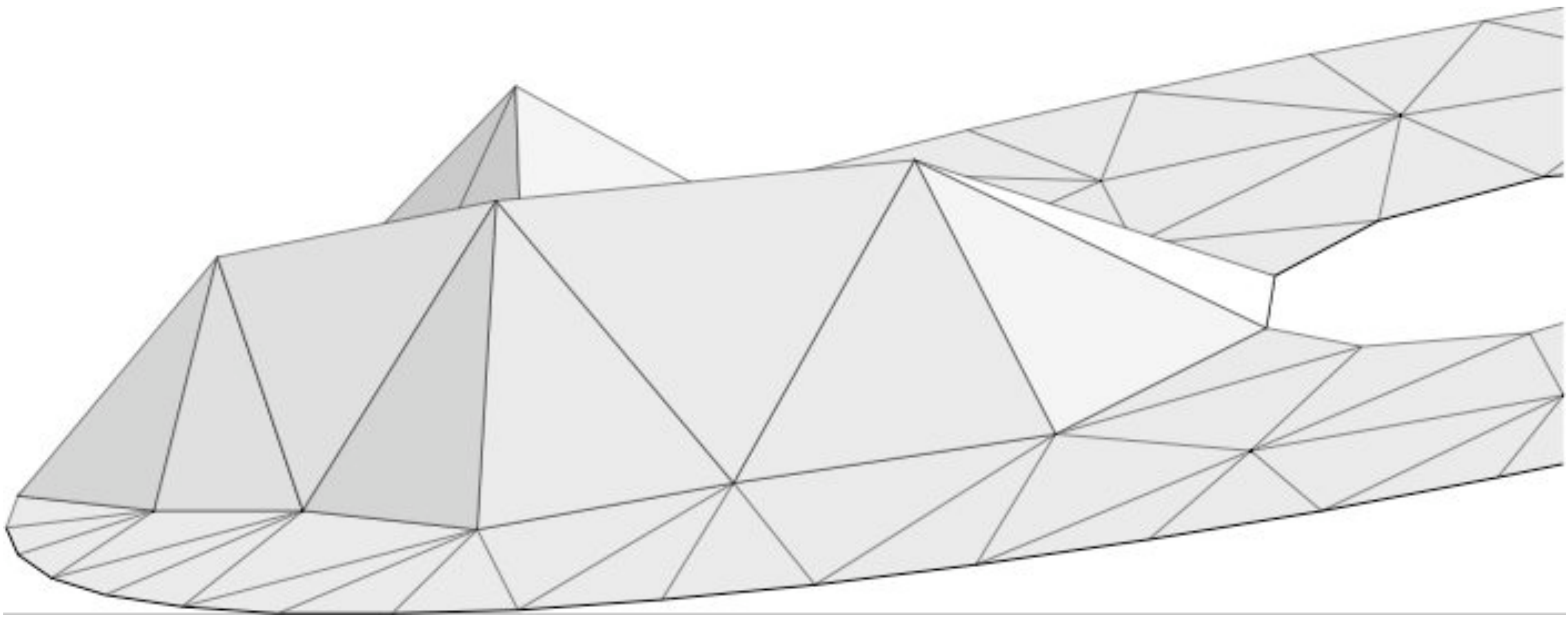
2-d Adaptive Mesh Generation

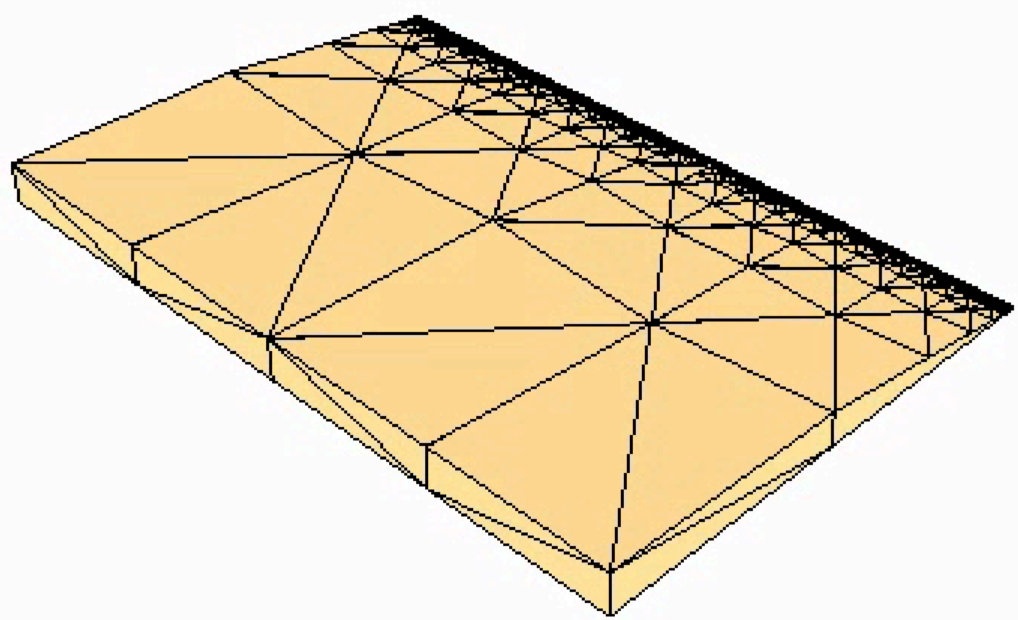


2-d Adaptive Mesh Generation

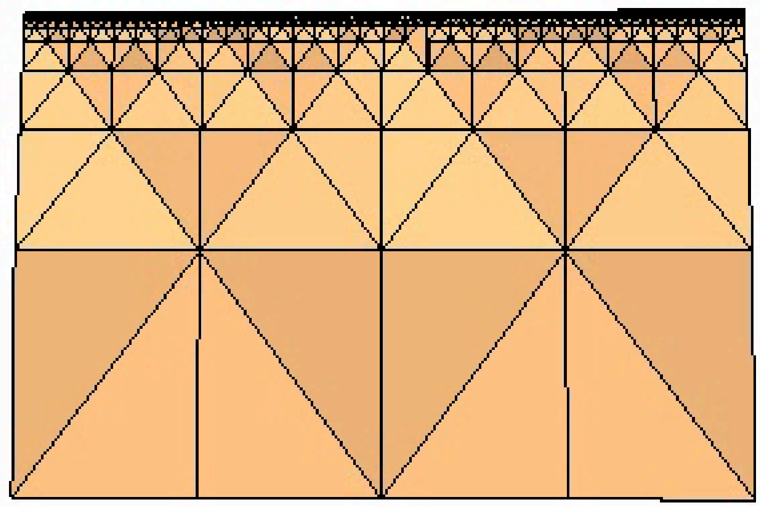


2-d Adaptive Mesh Generation





Courtesy: Shuo-Heng and Michael Garland



Courtesy: Shuo-Heng Chung and Michael Garland

SDG Is Time Consuming

- Some simulations would take days on a single processor.
- We want to parallelize it to speed up simulations!
- There are multiple ways of parallelizing it.
- A goal of the parallelization is to use existing frameworks where possible.

Master/Slave Parallelization of SDG

- The first parallelization of the SDG method was based on the observation that each patch could be solved independently.
- Thus the space mesh is not partitioned, but maintained on one master processor.
- Workers request patches to solve from the master processor.
- This method resulted in a bottleneck at the processor holding the entire space mesh.

Can We Parallelize the Geometry?

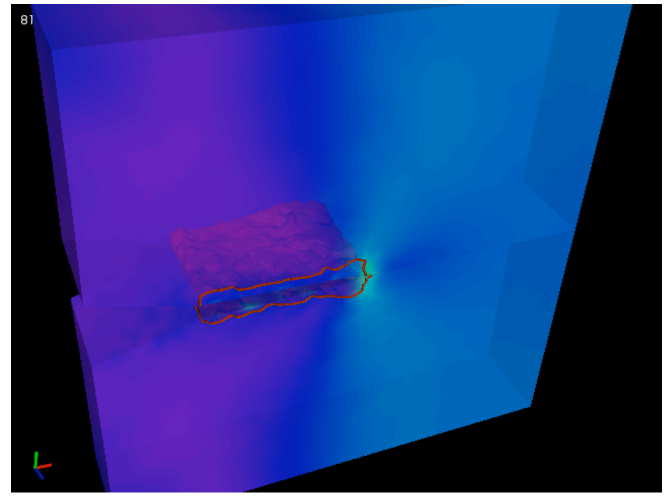
- Do not want a single processor bottleneck.
- We have an initial mesh, we'll partition the geometric space mesh.
- We need consistent ghost element layer.
- We need a locking mechanism for updating ghost values at appropriate times to ensure we have a consistent mesh.
- We will need the ability to incrementally add/remove elements to/from the mesh, maintaining consistency across all processors.

Parallel Frameworks for Unstructured Meshes:

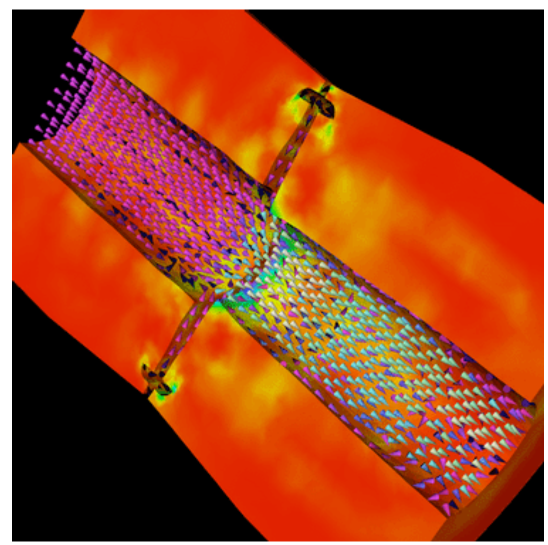
- ParFUM (Parallel Programming Lab, UIUC)
- Sierra(Sandia National Labs)
- Simmetrix
- Roccom(Center for Simulation of Advanced Rockets, UIUC)
- SUMAA3d(Argonne National Laboratory)
- UG

ParFUM Existing Features
(Parallel Framework for Unstructured Meshes)

- Originally designed for standard structural dynamics codes
- Extended to support Fluid Dynamic codes(Finite Volume)
- Local element/node ID numbering
- Efficient ID translation for communicating
- Partitioning
- Ghost layer generation
- Field registration and updating for shared nodes, ghosts



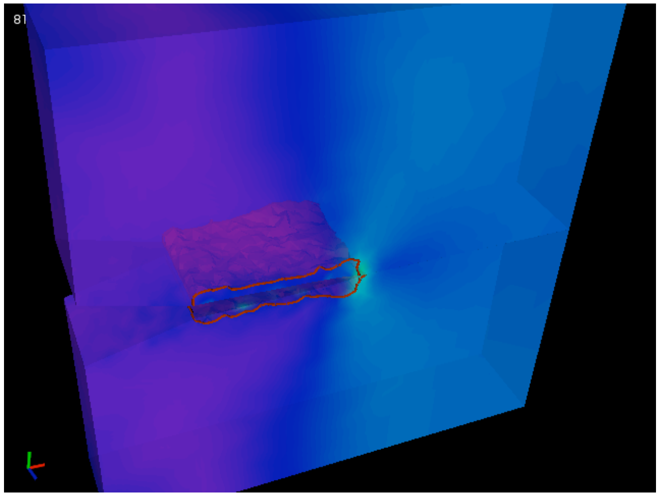
3-D Fractography in FEM



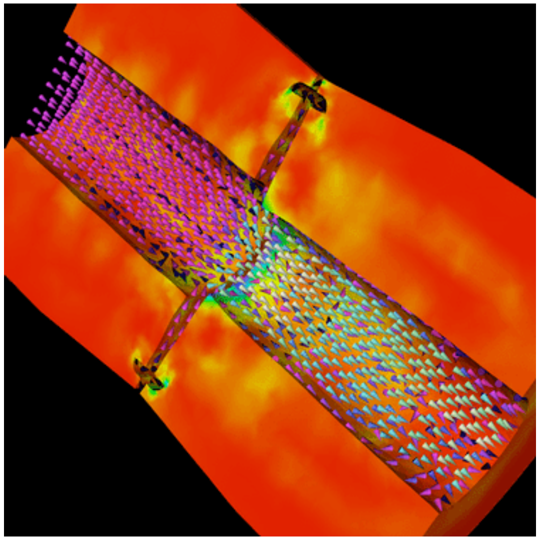
Rocket Burn Simulation, CSAR

ParFUM Existing Features
(Parallel Framework for Unstructured Meshes)

- ParFUM programs look similar to serial codes, operating upon local elements/nodes and ghost layers
- Can write programs in Fortran, C, C++
- Visualization Tools
- Collision Detection Library
- Tet Data Transfer Library



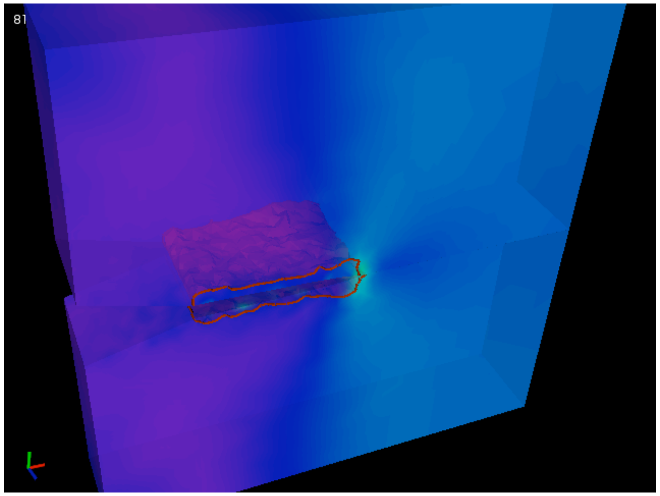
3-D Fractography in FEM



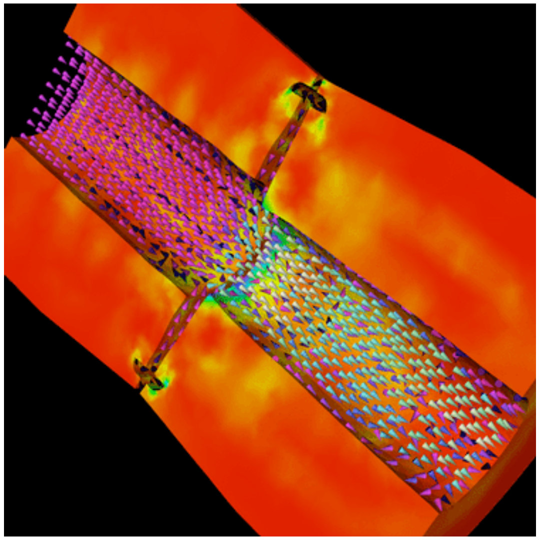
Rocket Burn Simulation, CSAR

ParFUM Existing Features
(Parallel Framework for Unstructured Meshes)

- •Virtualization
- •Load balancing(explicit or asynchronous)
- •Fault tolerance
- •Checkpointing
- •Performance Analysis



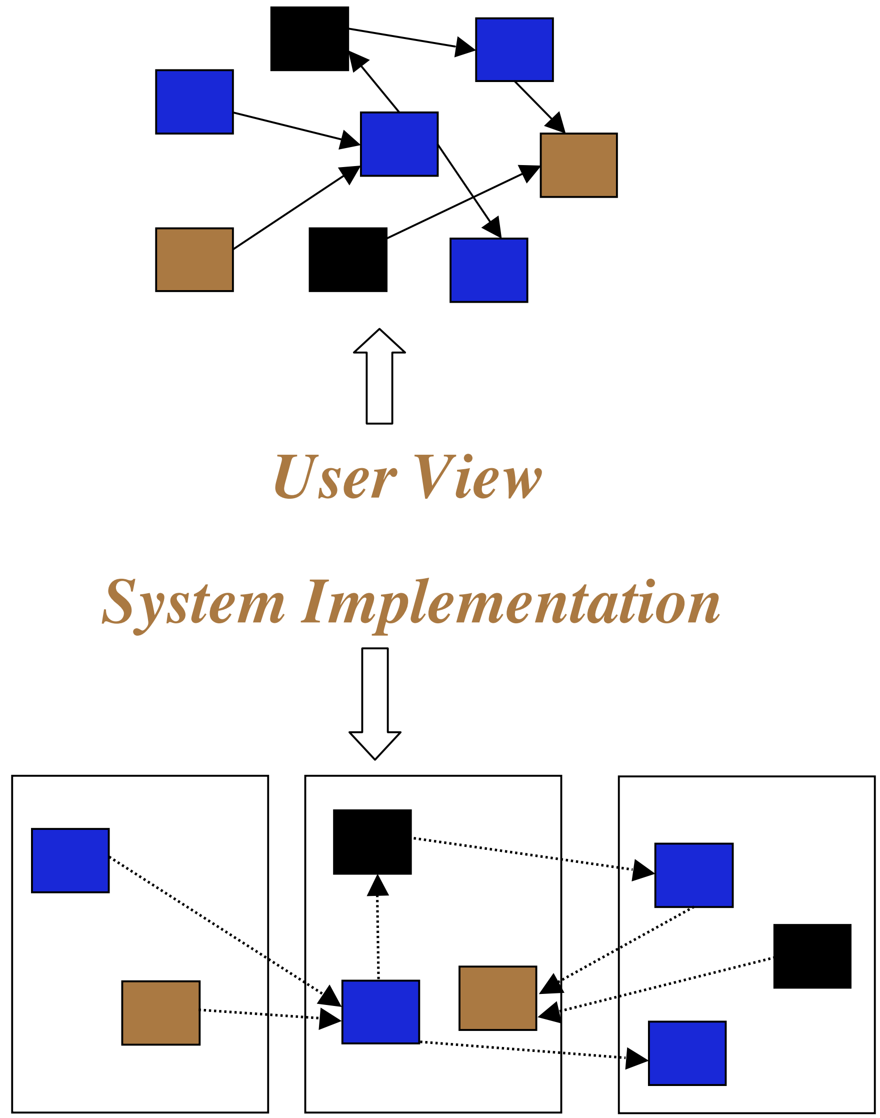
3-D Fractography in FEM



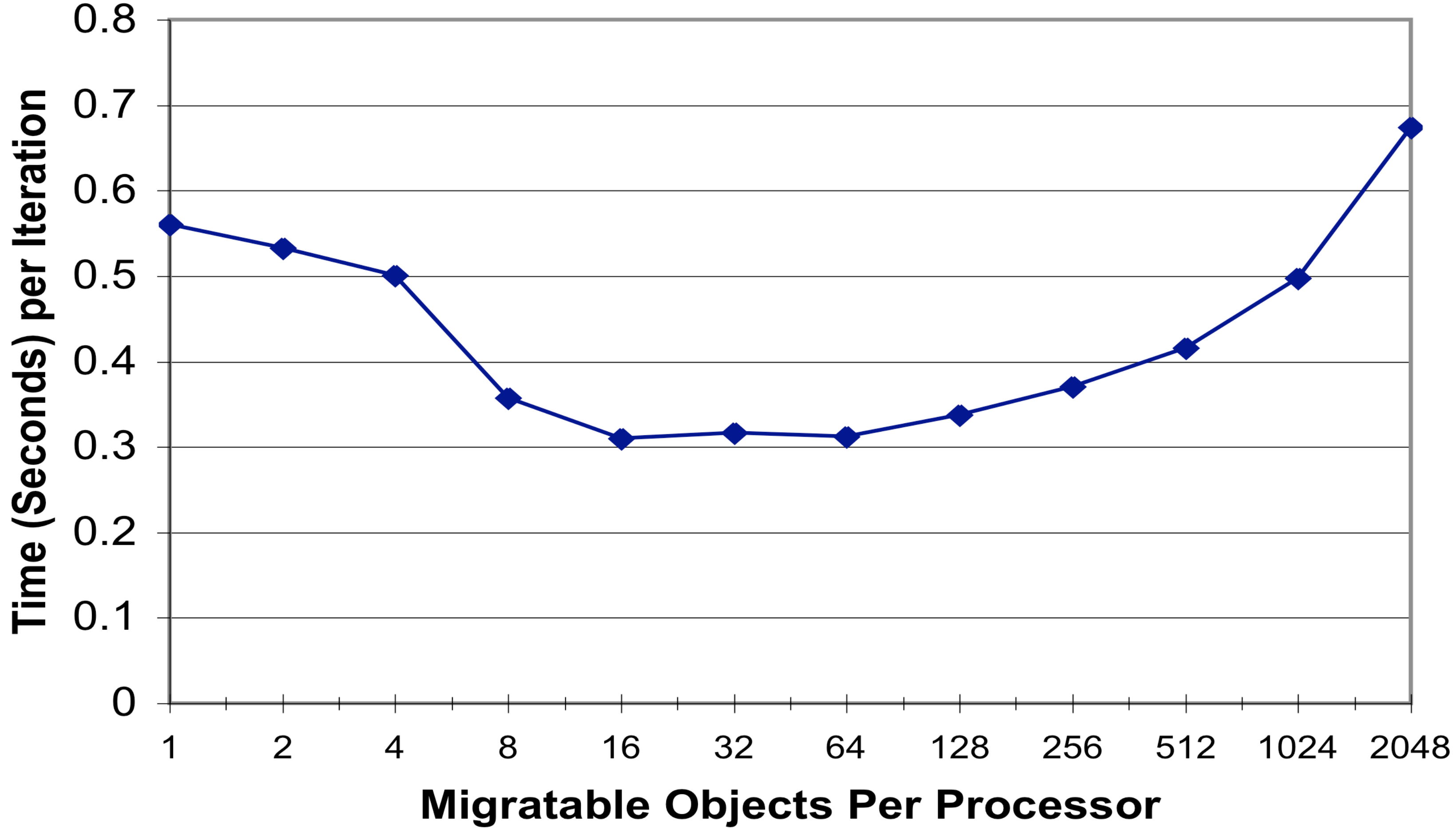
Rocket Burn Simulation, CSAR

Virtualization

- Charm++ Runtime System
  - Applications built using **migratable objects**
- Virtualization = **multiple migratable objects per processor**
- Load Balancing
  - Principle of Persistence
- High(90-100%) Processor Utilization and Scaling.
- Automatic Checkpointing
- Each ParFUM mesh chunk mapped to a migratable object.



Benefit of Virtualization to Structural Dynamics Application



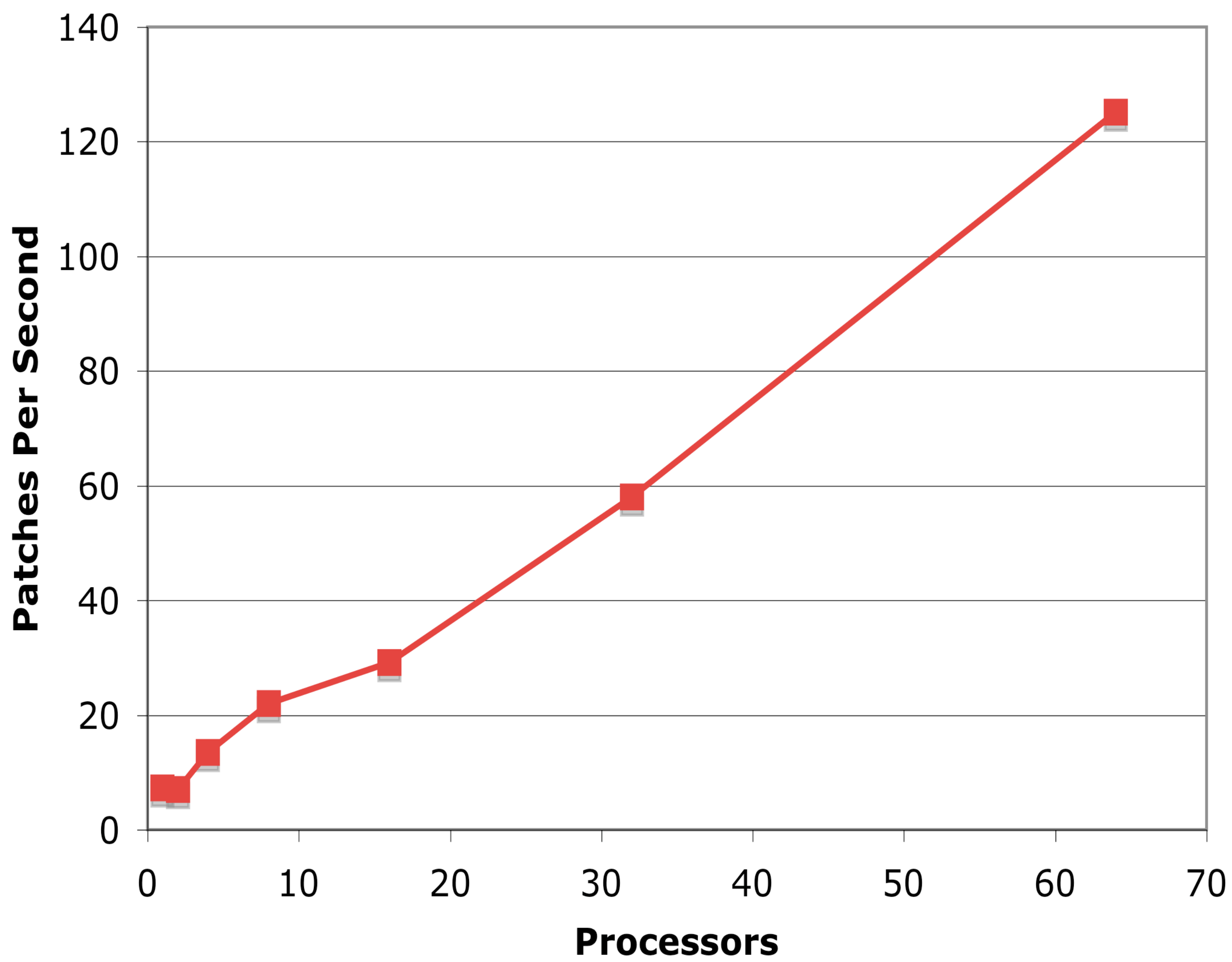
ParFUM Application
On Eight Physical Processors

ParFUM - New Features

Required for non-adaptive space-time meshing

- *Incremental* updates to ghost layers of adjacent processors
- *Locking* of individual elements or nodes.
- No global synchronization.
- New adjacency data structures
  - Element to element
  - Node to node
  - Node to element

Non-adaptive SDG Program
Initial Results



ParFUM Support For Adaptivity

- Access to general-purpose mesh modification primitives
- Mesh Refinement
- Mesh Coarsening

ParFUM - New Features

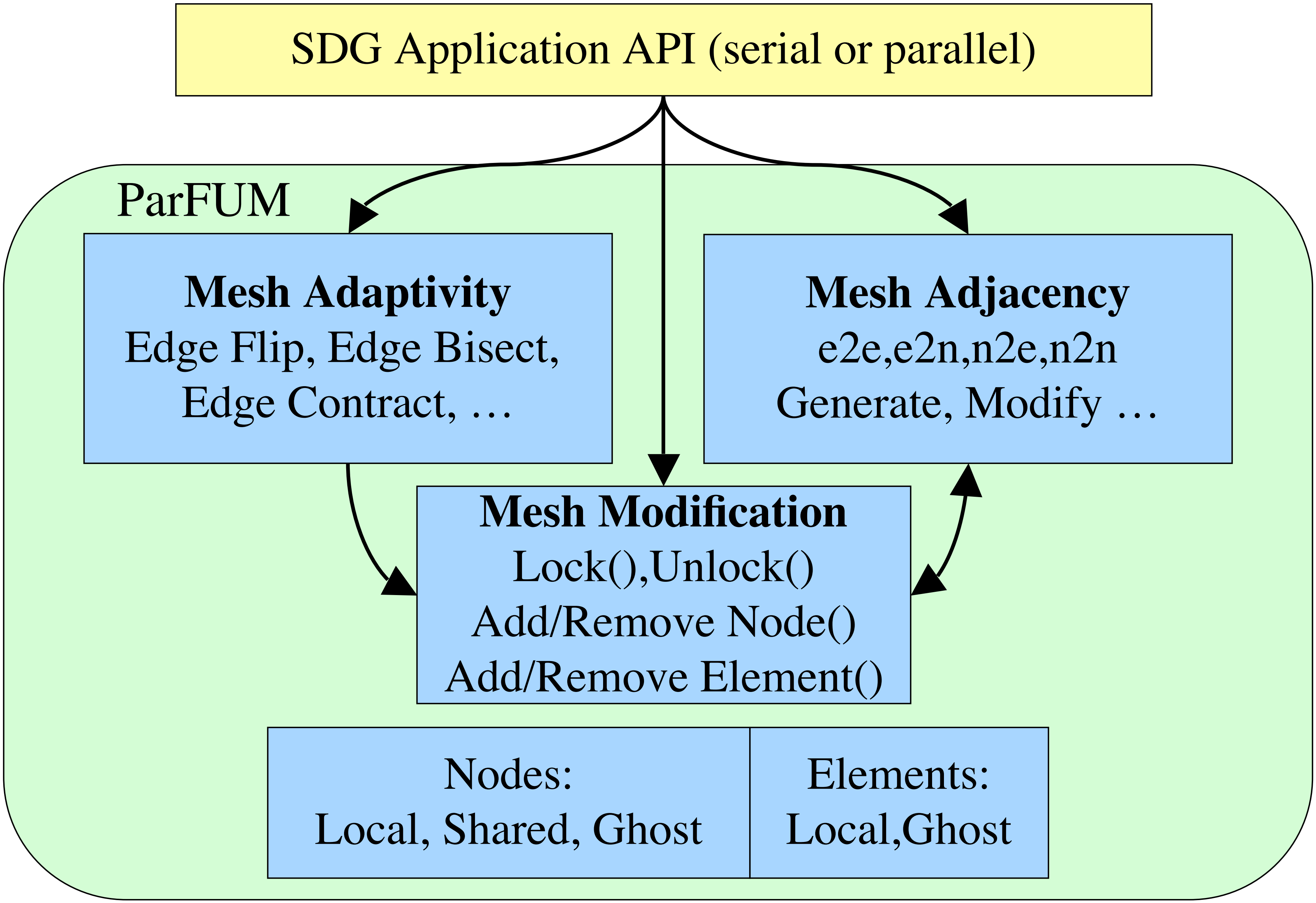
Useful for adaptive space-time meshing

- Current work
- Non-trivial challenges for a framework which doesn't allow unstructured meshes to be modified asynchronously during execution.
- Must maintain consistent mesh across all processors, with correct ghost layers, shared nodes, ghost nodes, and adjacencies at any time.

ParFUM Support For Adaptivity

- Load balancing is required in any efficient framework for adaptive SDG, since mesh partitions can differ in size by orders of magnitude.
- We have already extended ParFUM to provide parallel incremental mesh modification primitives.
- The primitives allow simple coding of incremental mesh modification, refinement, coarsening, and repair routines.

ParFUM Structure



Mesh Modification Examples

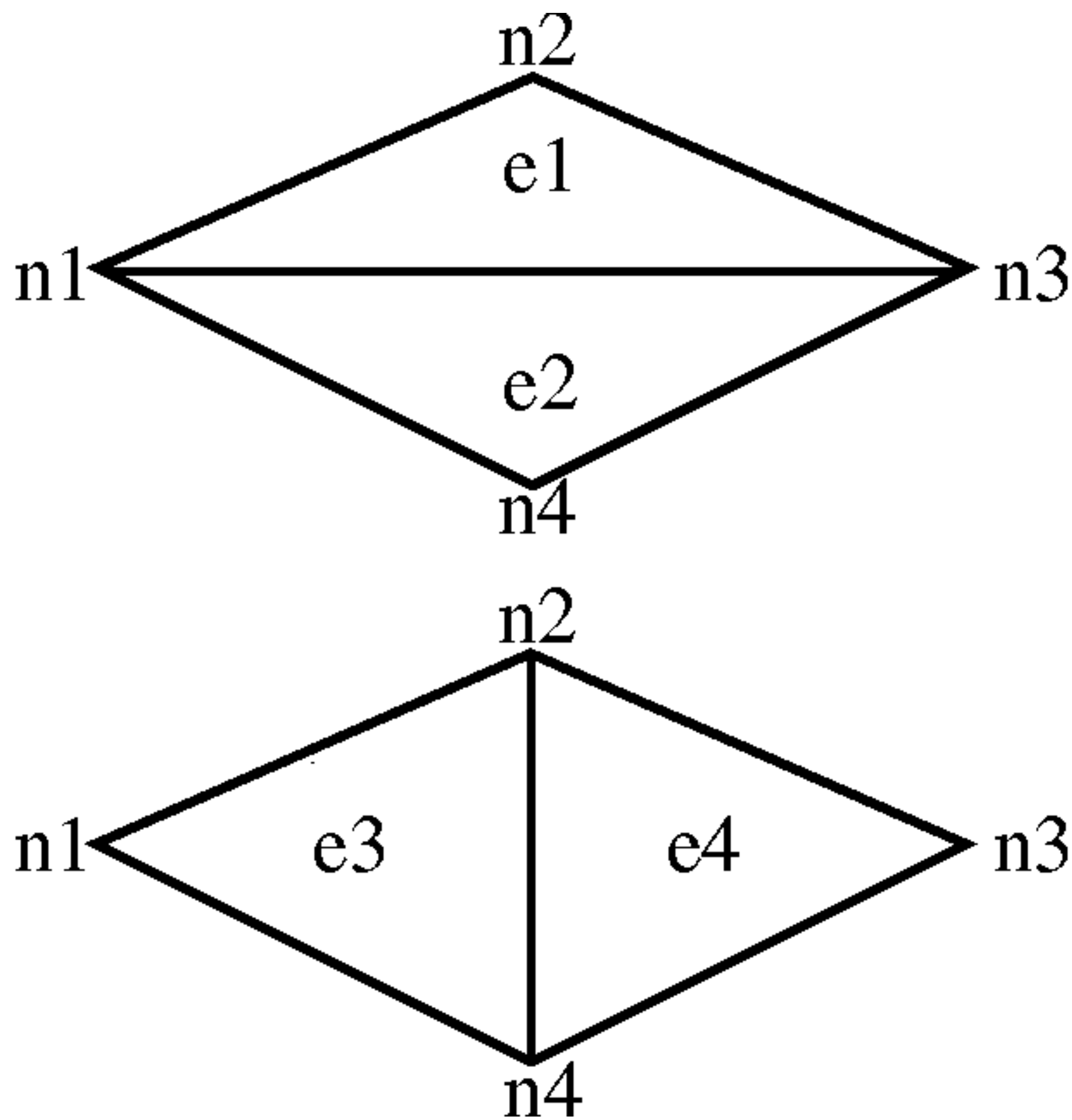
Edge Flip:

Remove elements e1

Remove element e2

Add element (n1,n2,n4)

Add element (n2,n3,n4)



Mesh Modification Examples

Edge Bisect:

Remove elements e1

Remove element e2

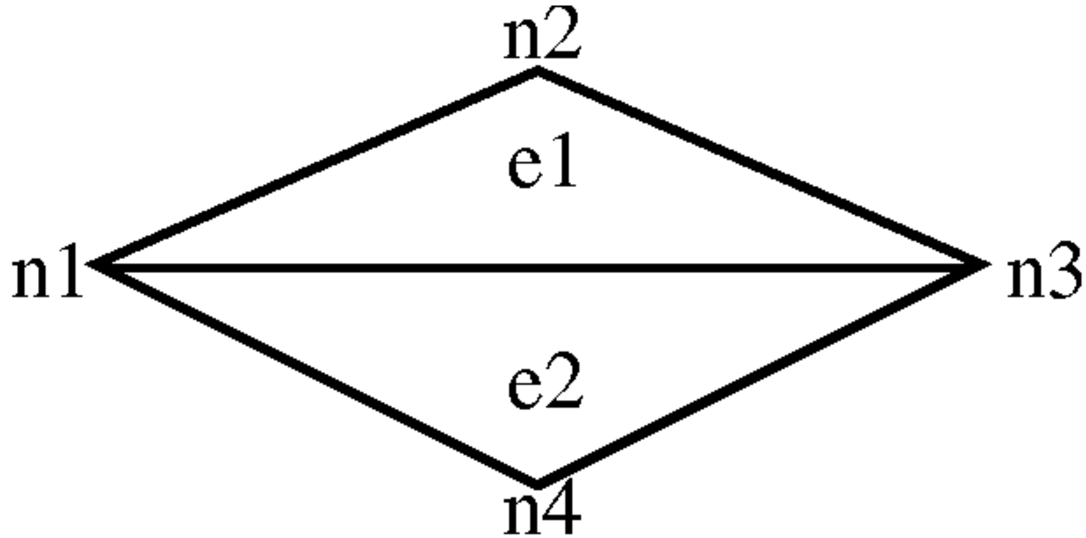
Add node

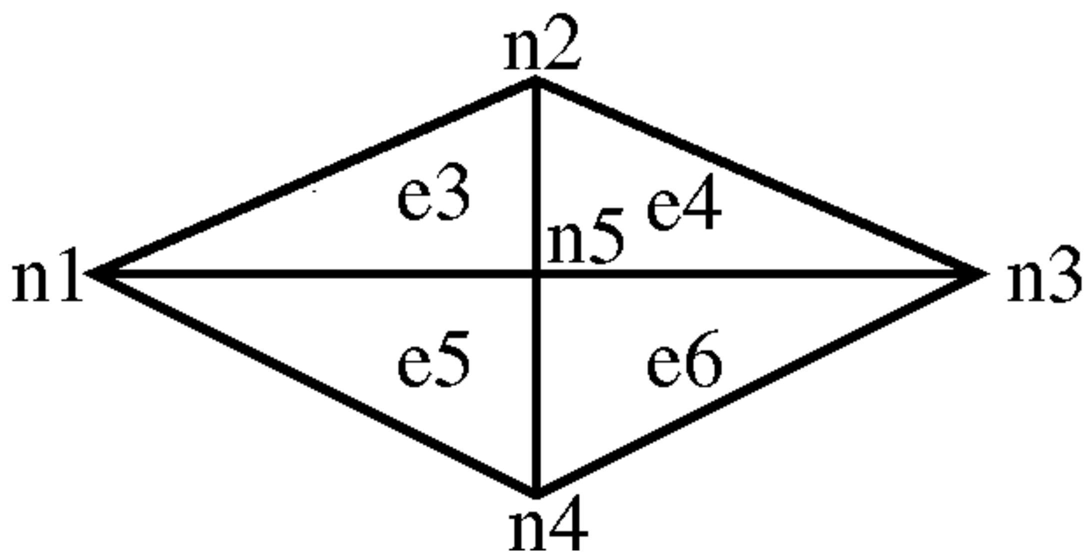
Add element (n1,n2,n5)

Add element (n3,n5,n2)

Add element (n4,n5,n3)

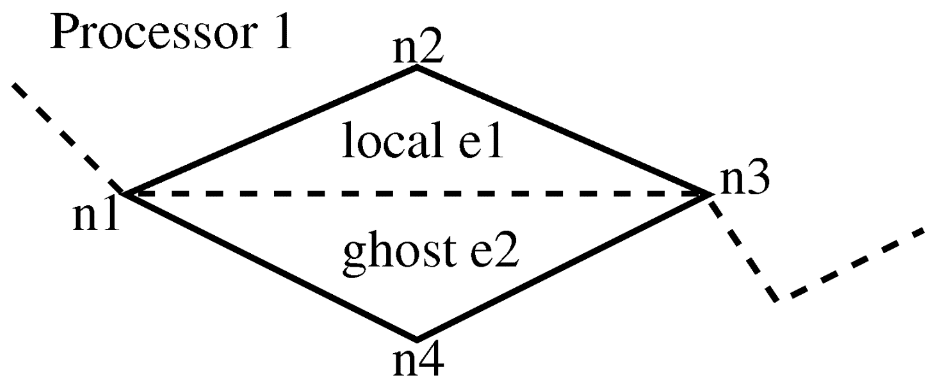
Add element (n4,n1,n5)



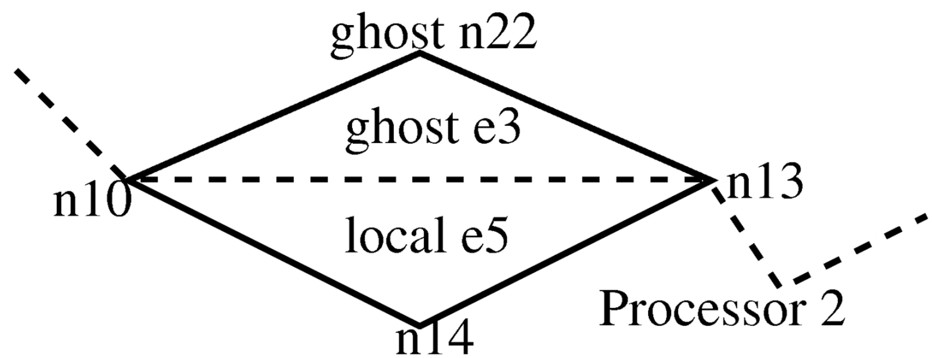


Mesh Modification in Parallel

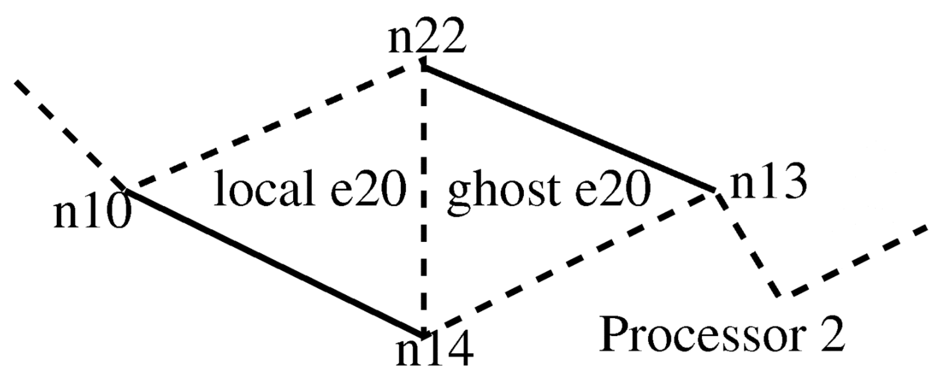
Mesh on Processor 1
before edge flip



Mesh on Processor 2
before edge flip



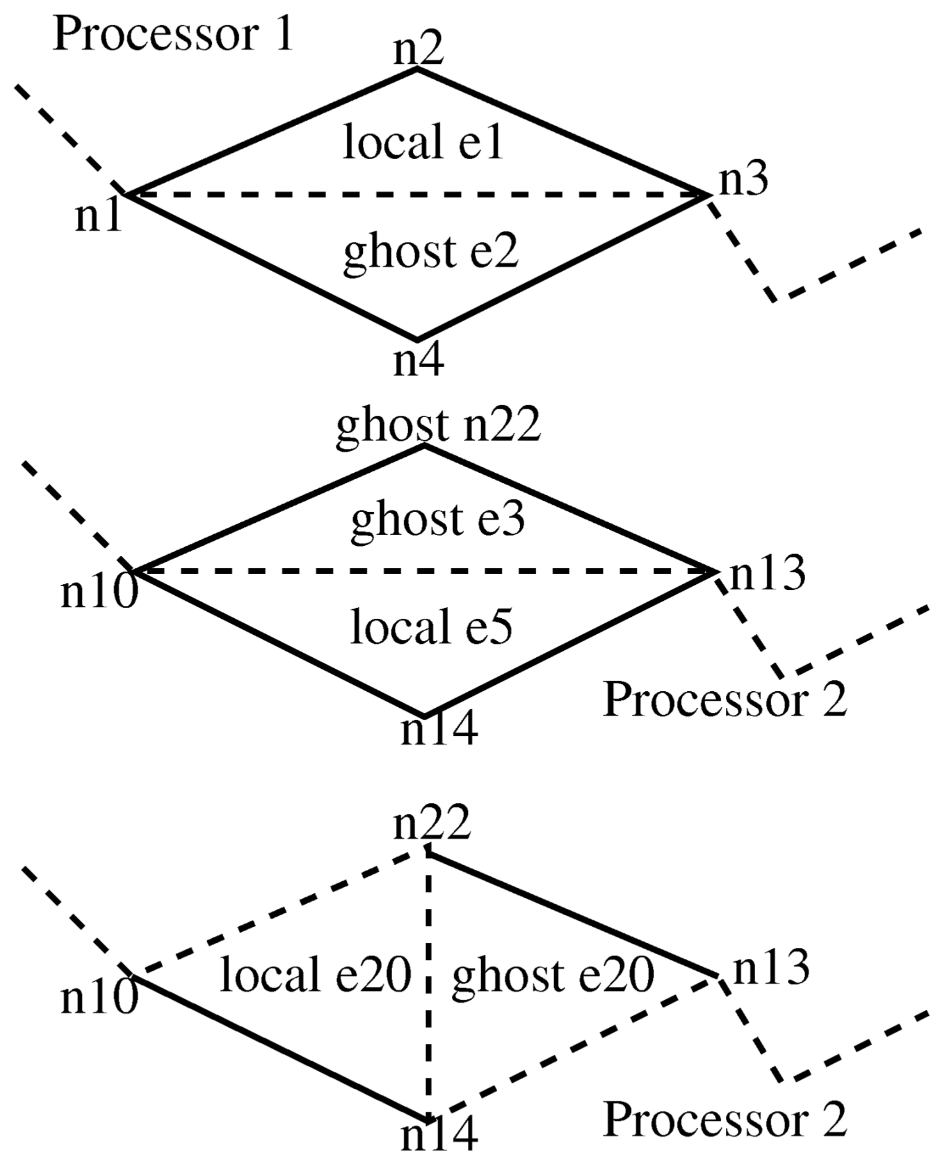
Mesh on Processor 2
after edge flip



Mesh Modification in ParFUM

Primitive Operations must do the following:

- Perform the operation on local and all applicable remote processors
- Convert local nodes to shared nodes when they become part of the new boundary
- Update ghost layers(nodes and elements) for all applicable processors. The ghost layers can grow or shrink



Thank-you for listening to my talk!

Questions or Comments?