Load Balancing Techniques for Asynchronous Spacetime Discontinuous Galerkin Methods

Aaron K. Becker (abecker3@illinois.edu)
Robert B. Haber
Laxmikant V. Kalé

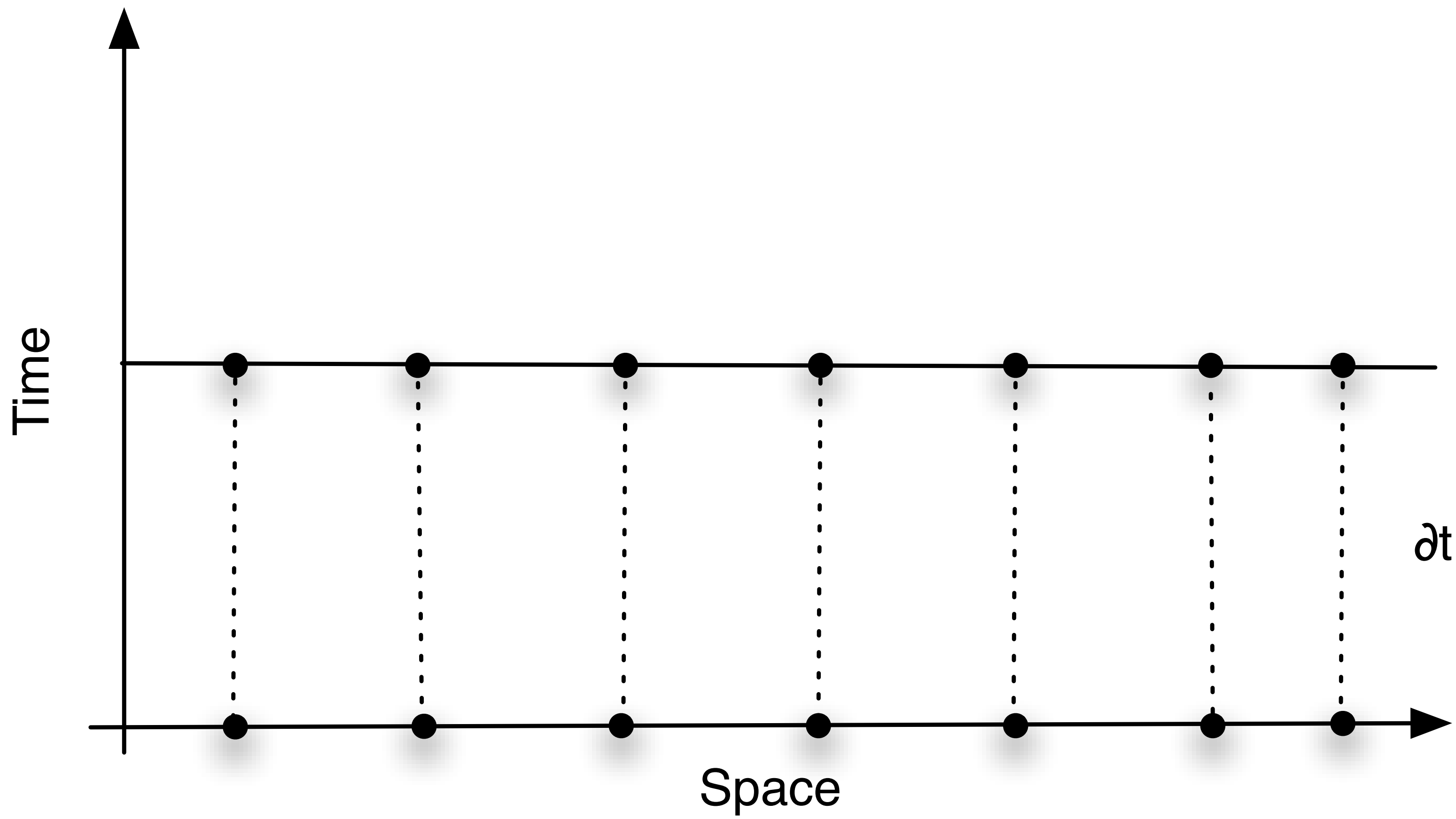
University of Illinois, Urbana-Champaign

Parallel Programming Lab
Center for Process Simulation and Design

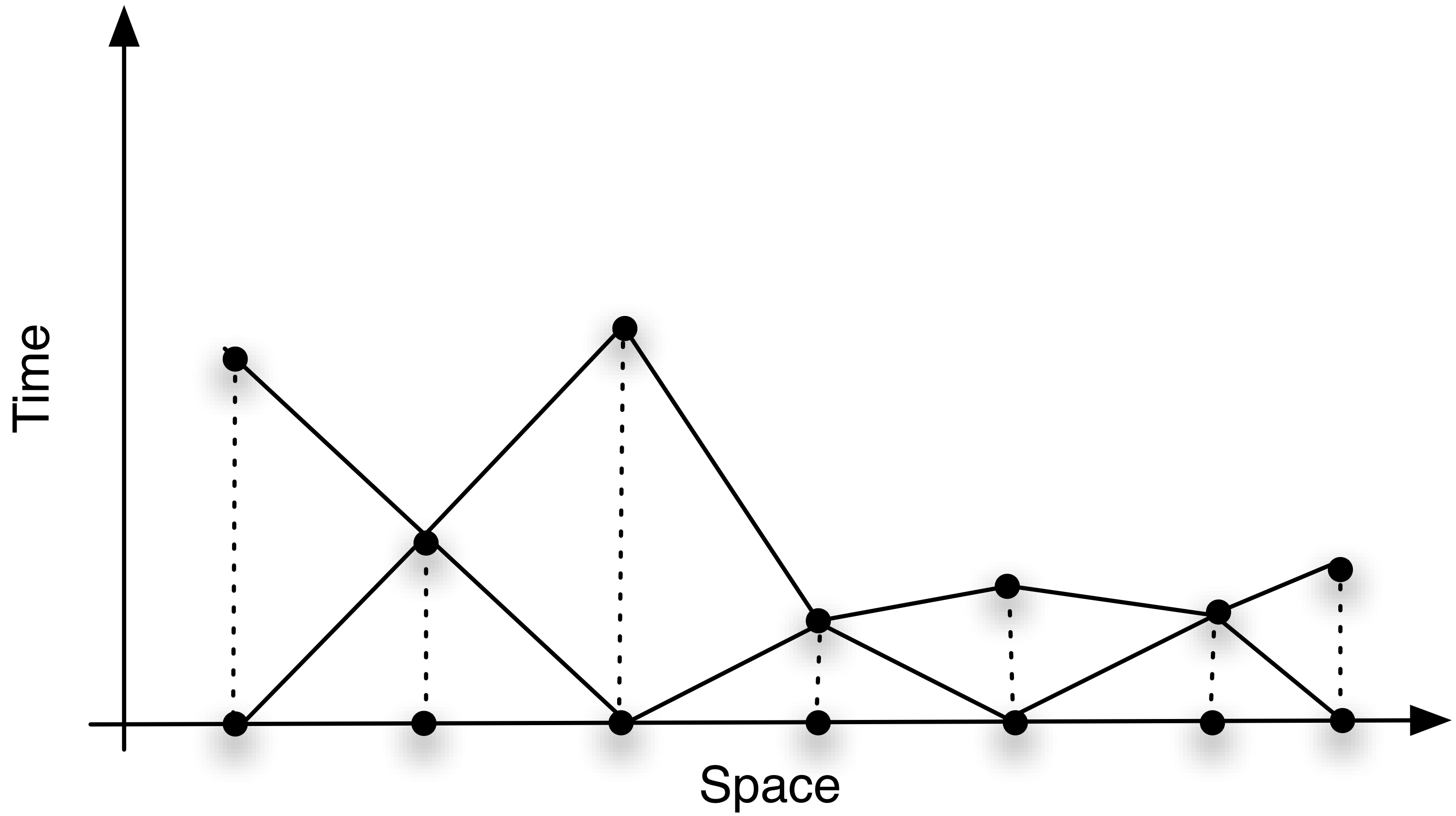
UNSCCM ’09

NSF: ITR/AP DMR 01-21695
ITR/AP DMR 03-25939

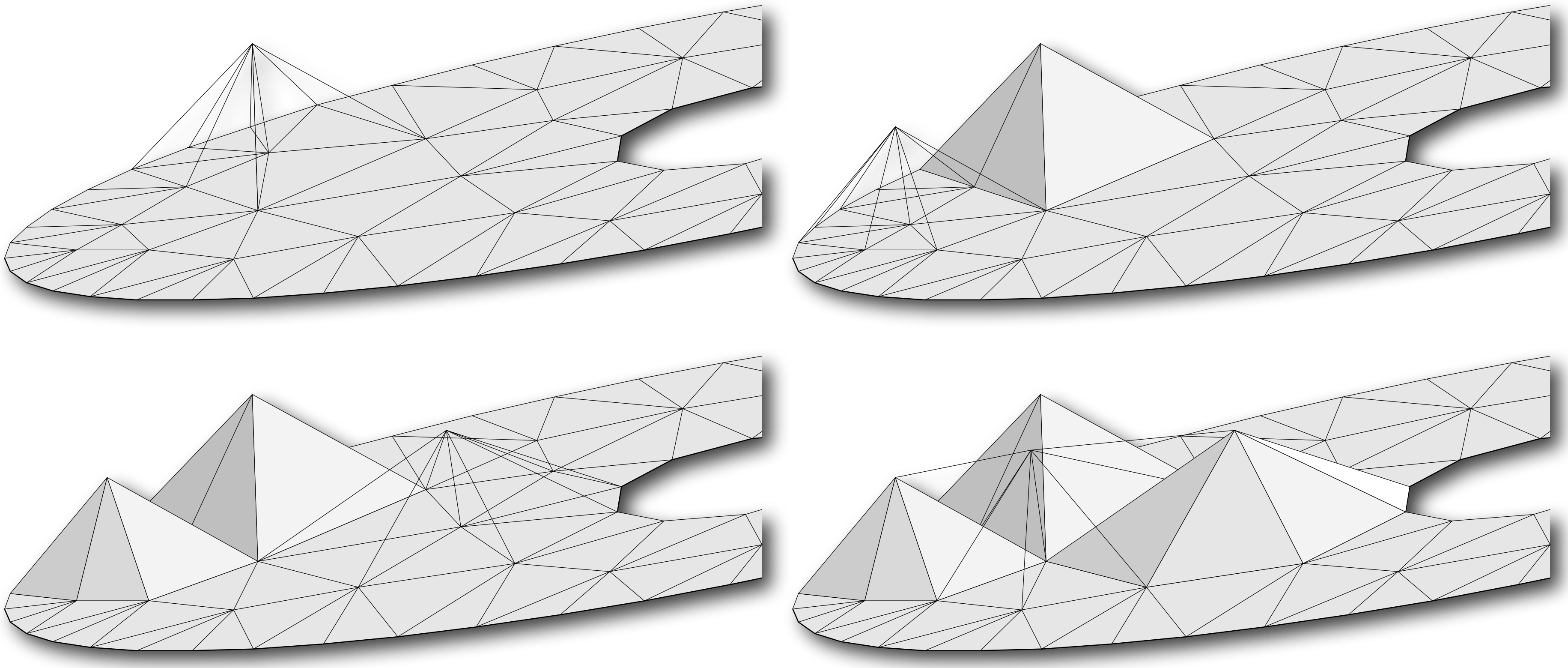
Fixed Timestep 1D Algorithm



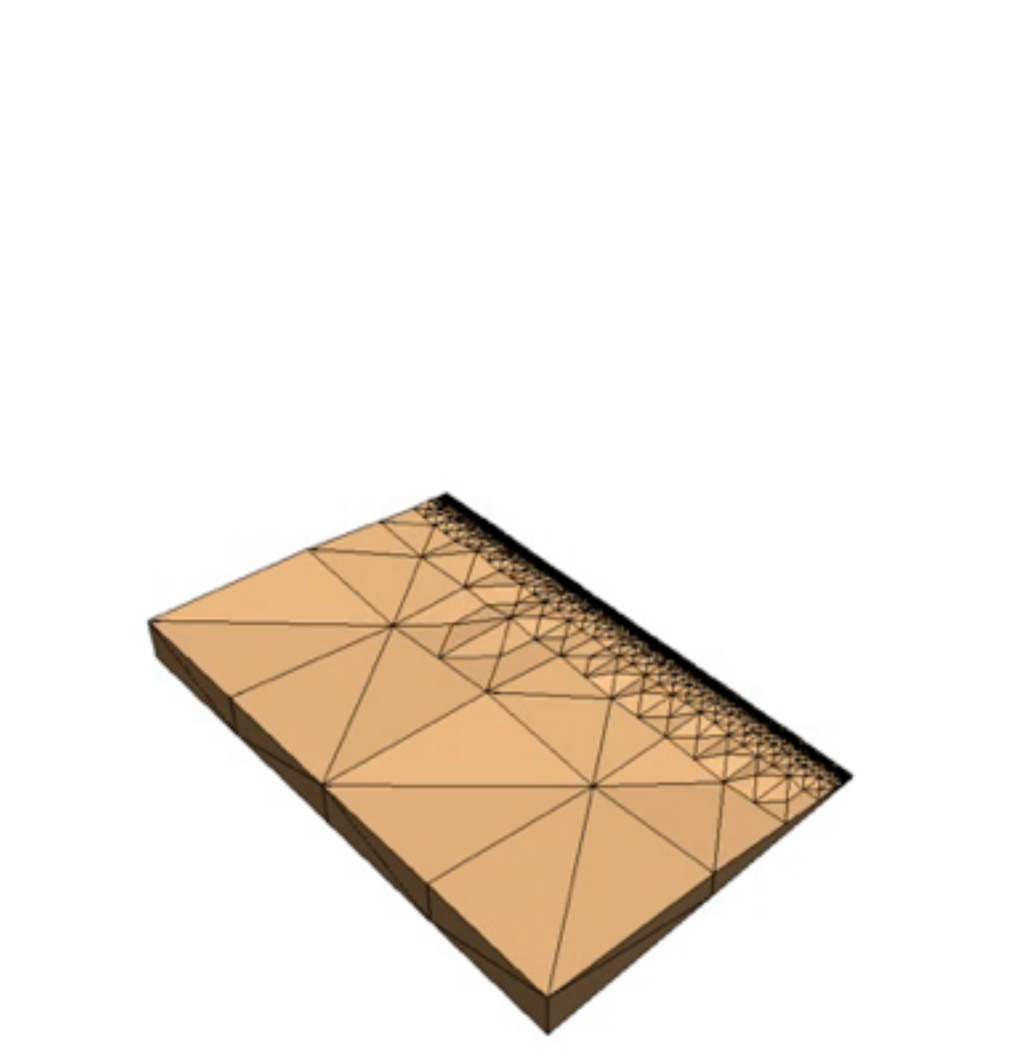
Tentpitcher: Causal Spacetime Mesh Advancing-Front Solution Strategy



Tentpitcher: patch by patch solution & meshing



Crack-tip Wave Scattering



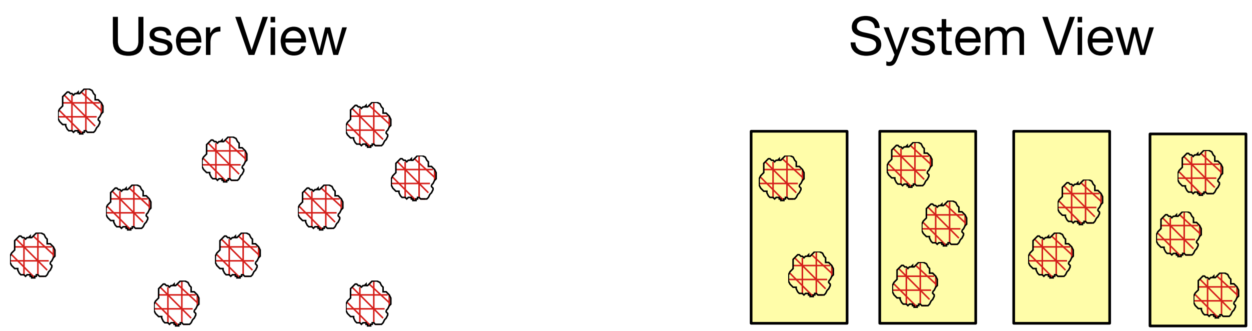
Parallelizing Tentpitcher

- Approach
  - take advantage of local decision-making algorithm to avoid global communication and promote scalability
  - build in latency tolerance to support large grain sizes

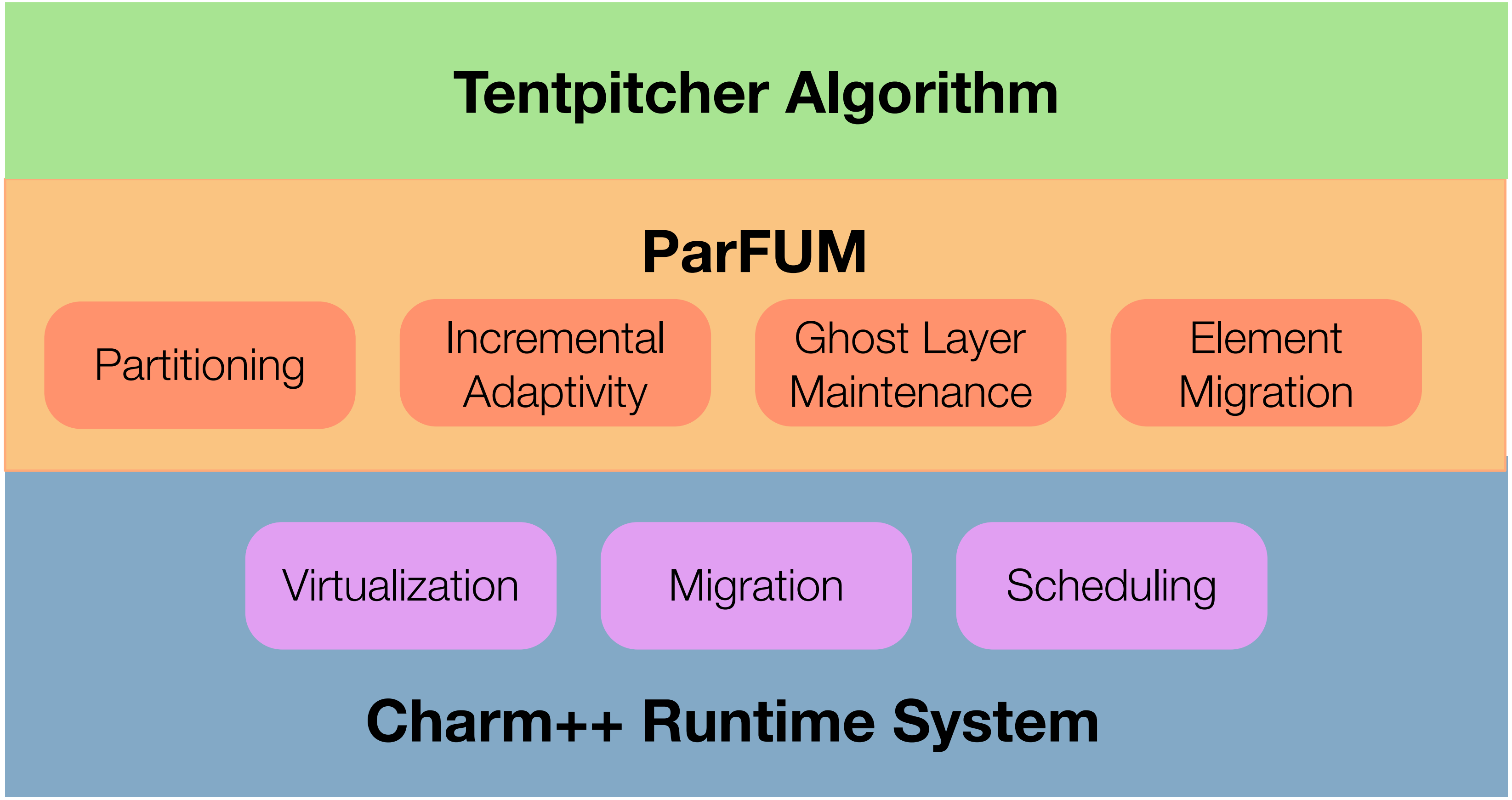
- Decompose and distribute space mesh
- All non-boundary operations are purely local
- Perform boundary communication on-demand using a message driven approach

Message-driven SDG

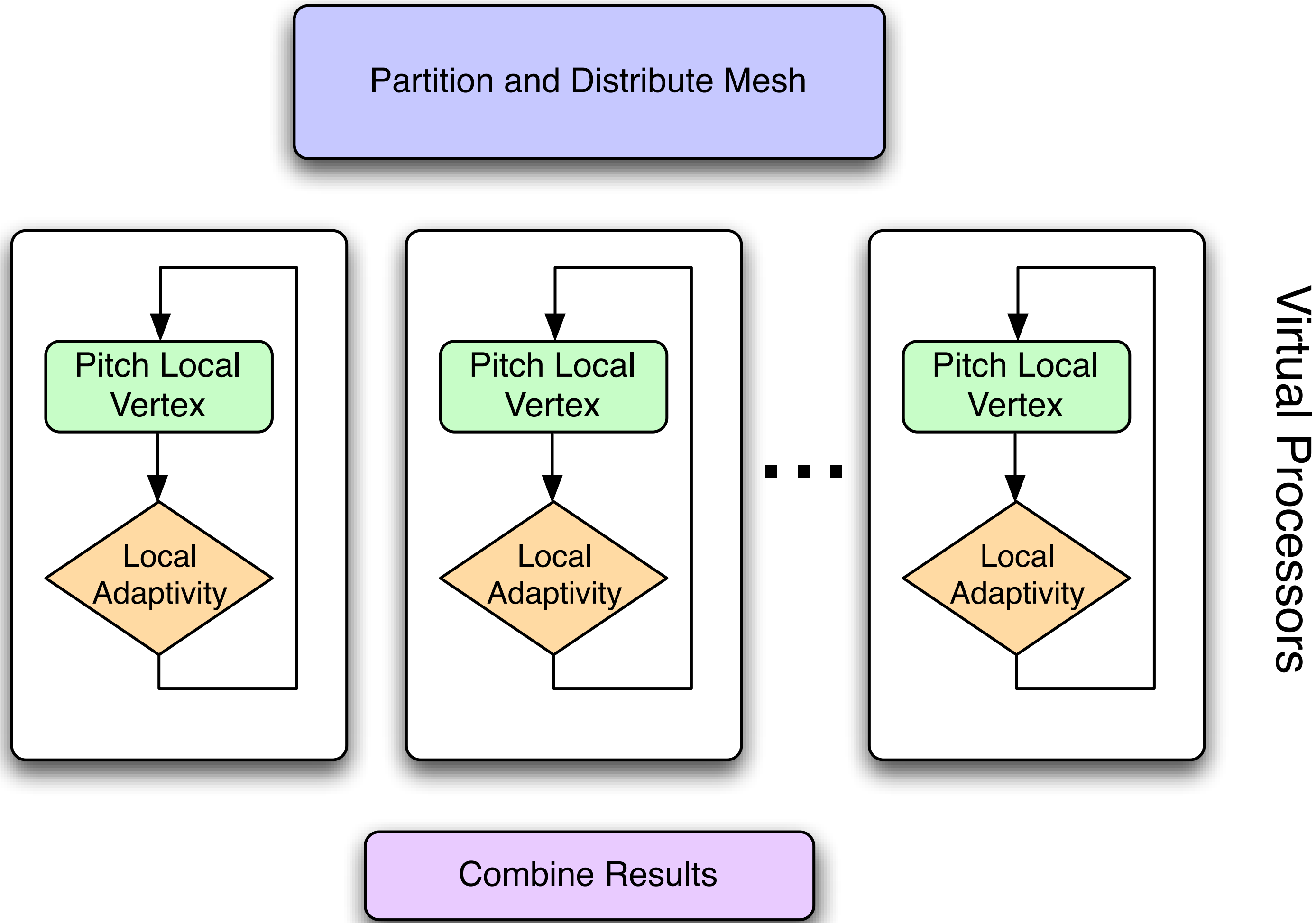
- Over-decomposition and virtualization
  - multiple mesh partitions per processor
  - computation on one partition can be overlapped with blocking communication on another local partition



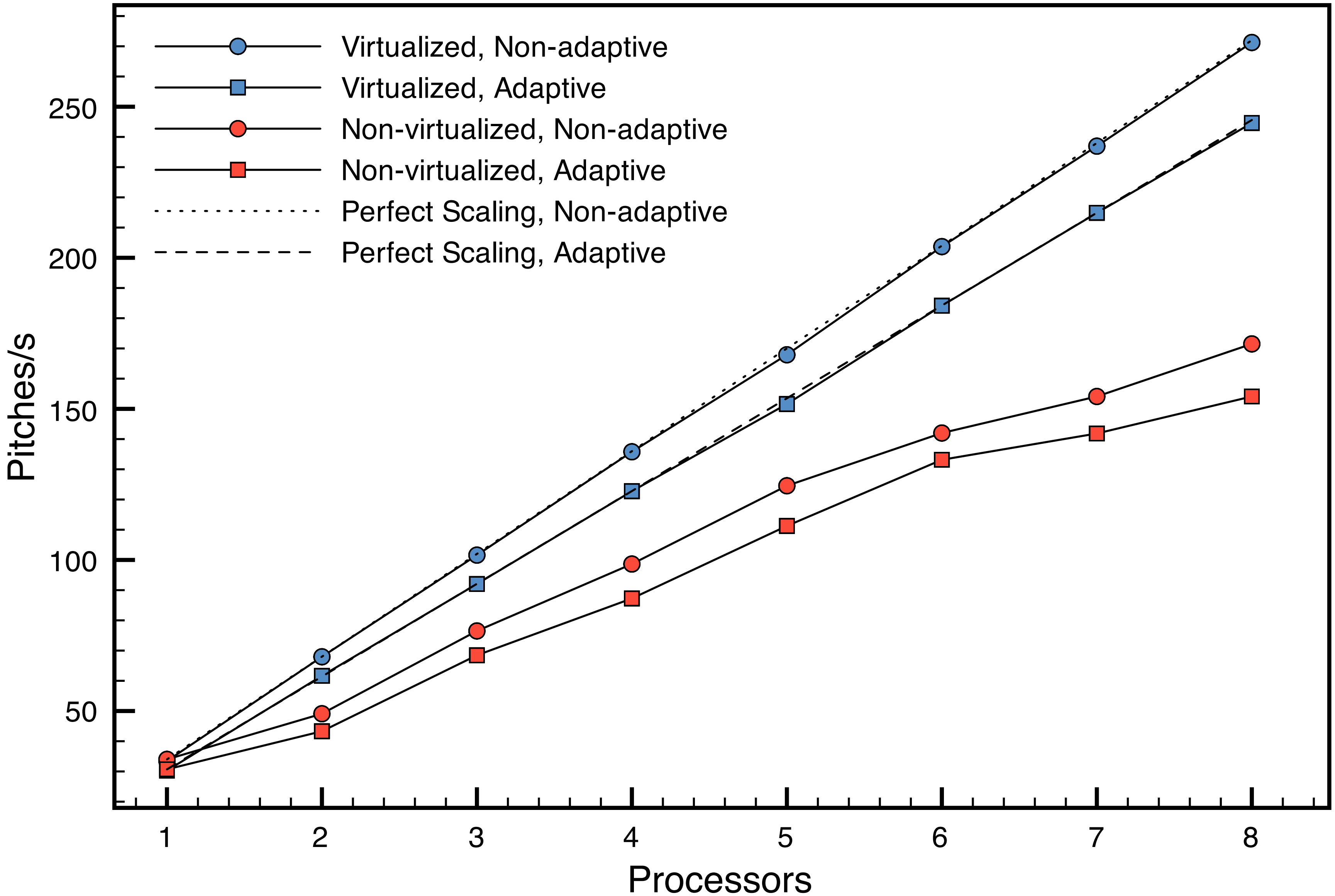
System Overview



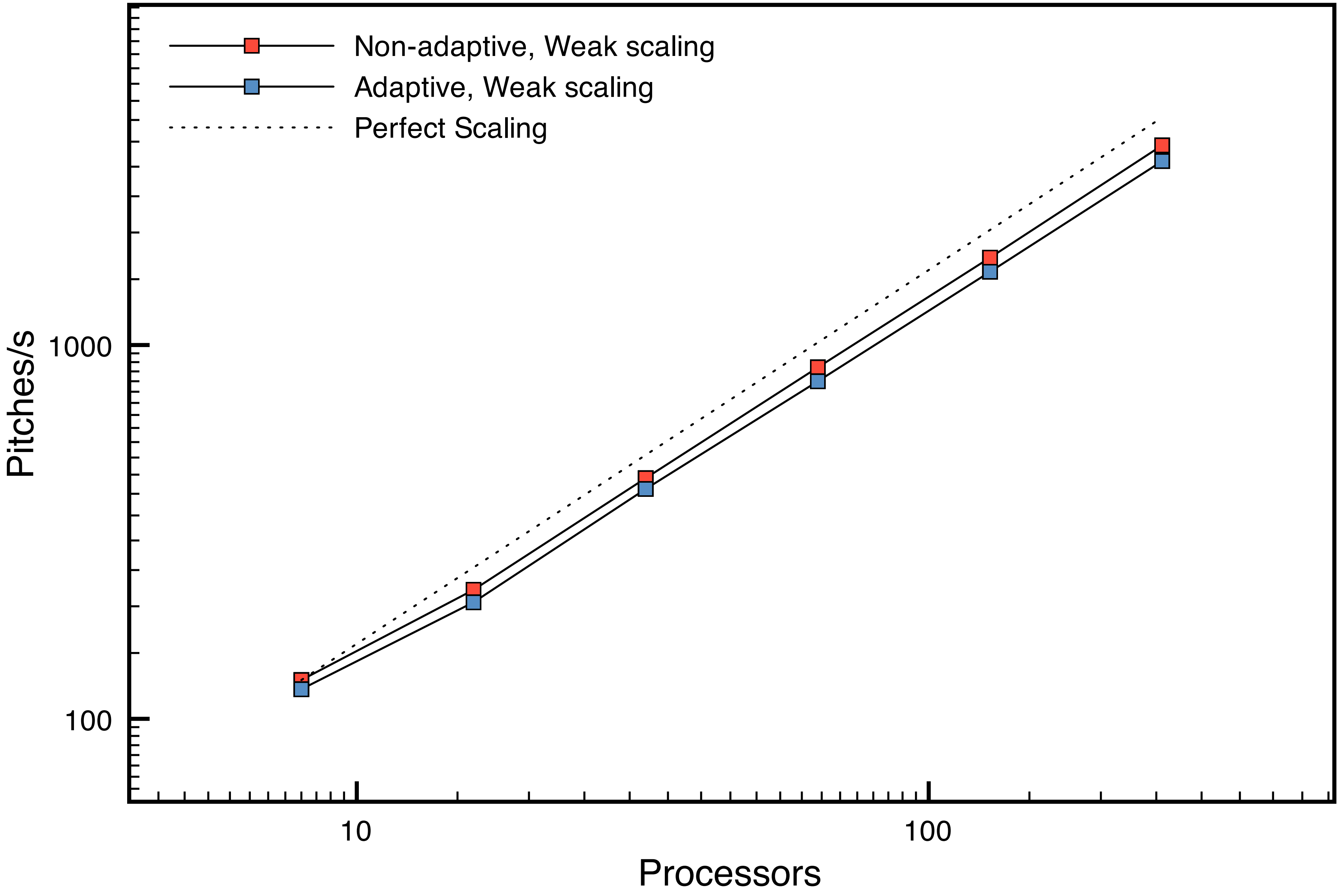
Code Structure



Performance Effects of Virtualization



SDG Cluster Performance (Abe)



Dealing with Load Imbalance

Aside from load imbalance, few barriers to scalability

This method naturally tolerates small imbalances

But, for some problems we expect large imbalances

Partition Migration

- Idea: take advantage of virtualization: there are multiple partitions per processor, so they can be rearranged to improve load balance
- Standard approach in virtualized environments: Charm++ supports a variety of algorithms for relocating partitions

  Advantages

    - built-in support, requires little modification of application
    - effective for moderate imbalances

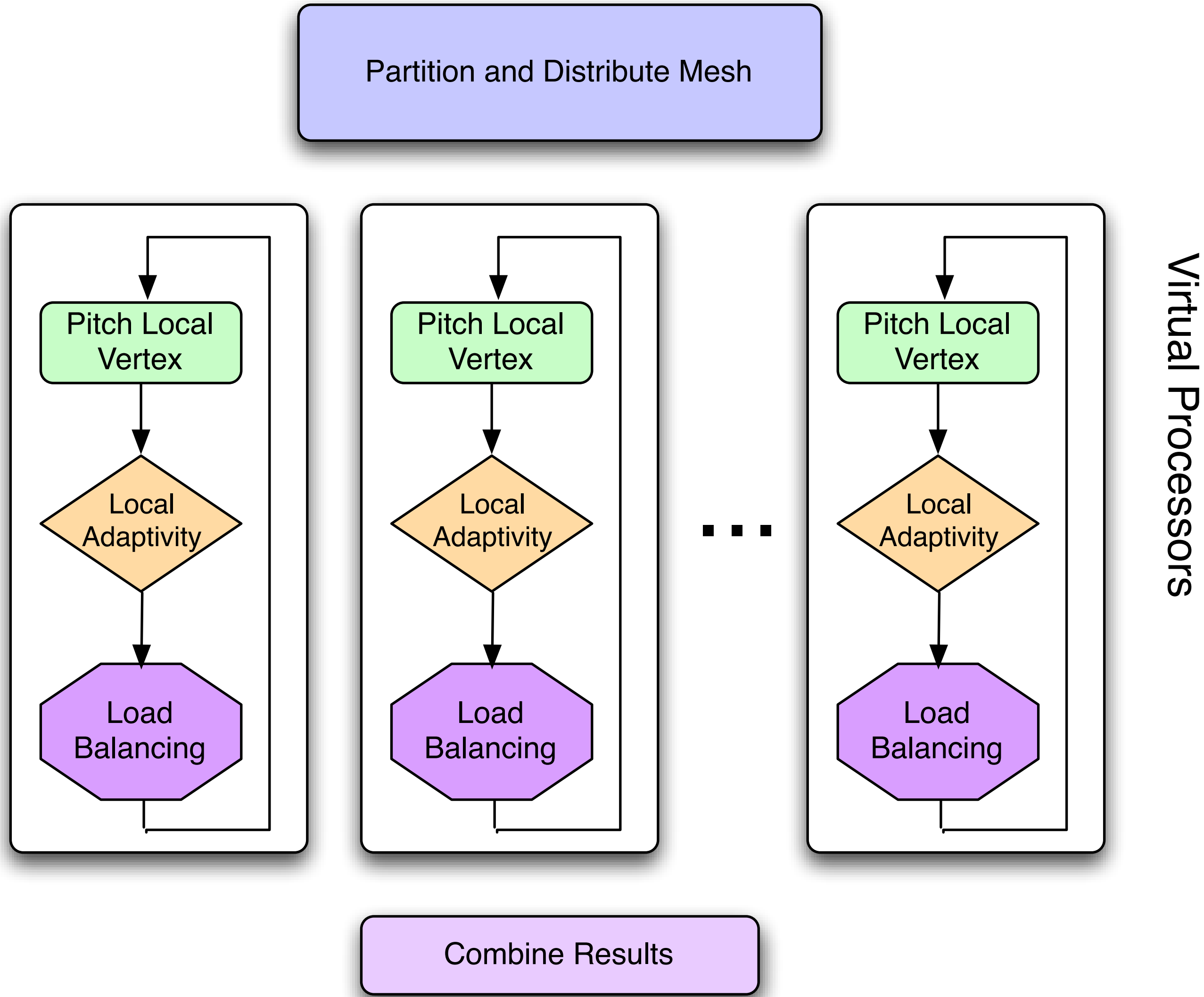
  Disadvantages

    - global, synchronous approach is a poor fit for tentpitcher
    - really large imbalances may not be fixable--the presence of dramatically overloaded partitions cannot be covered up without unacceptable overhead

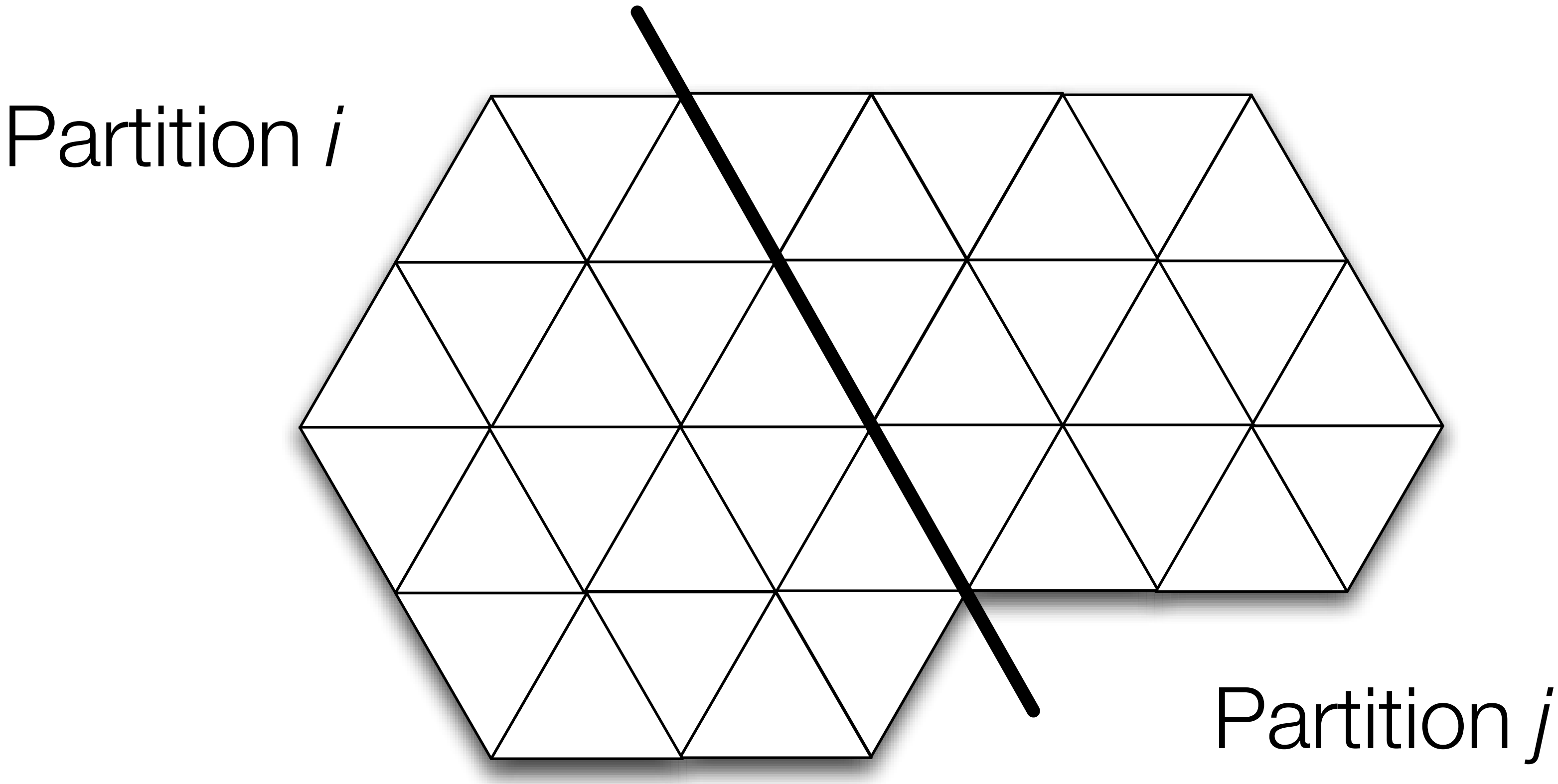
Diffusion Load Balancing

- Idea: apply purely local decision making process to load balance by migrating individual mesh elements across partition boundaries once load imbalance crosses a particular threshold value
- If neighboring partitions *i* and *j* have loads λ_i and λ_j, choose $r > 1$ and migrate elements from *i* to *j* when $r \lambda_i > \lambda_j$
- Advantages: requires only local synchronization and communication

Code Structure

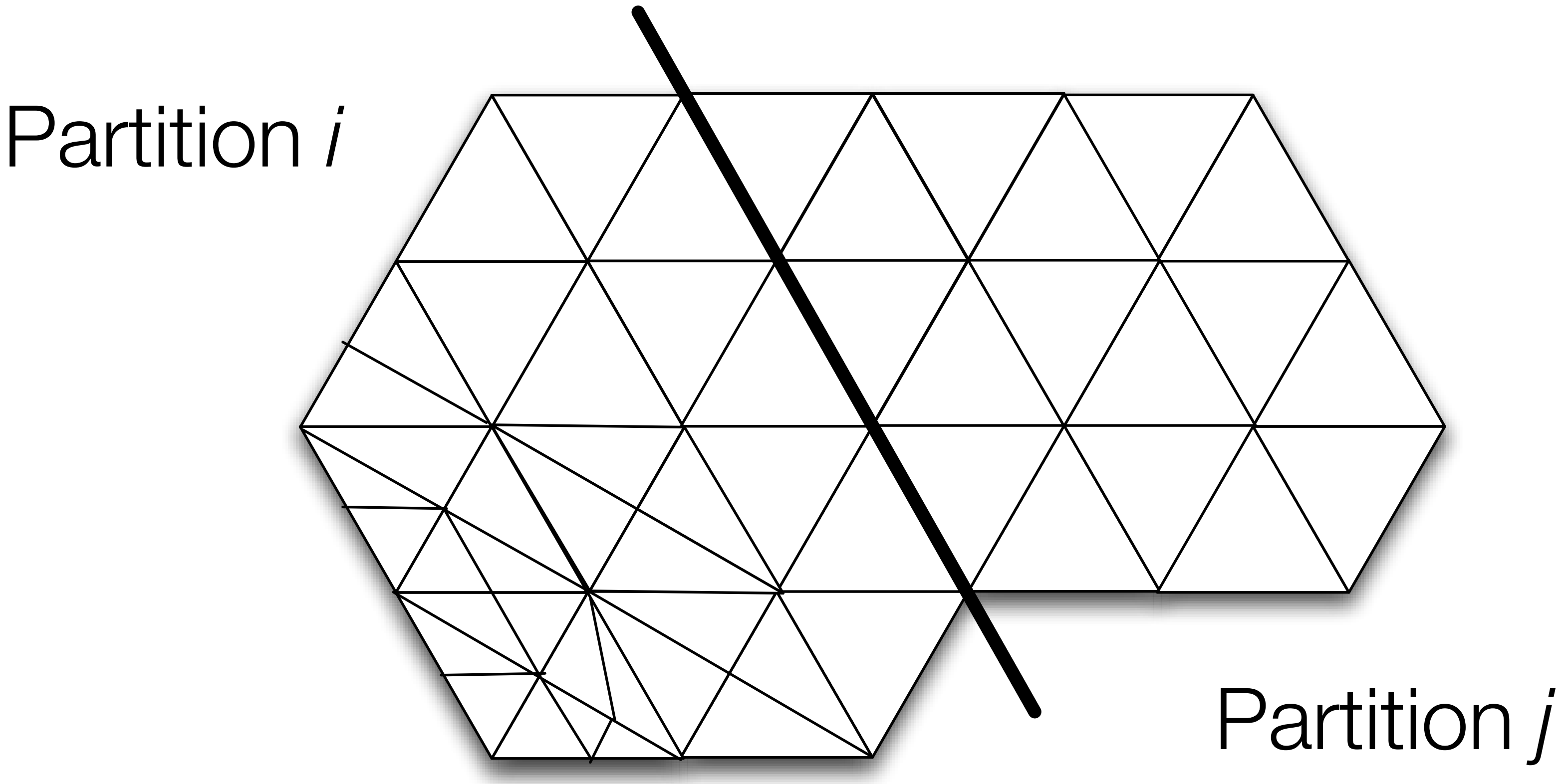


Diffusion Load Balancing



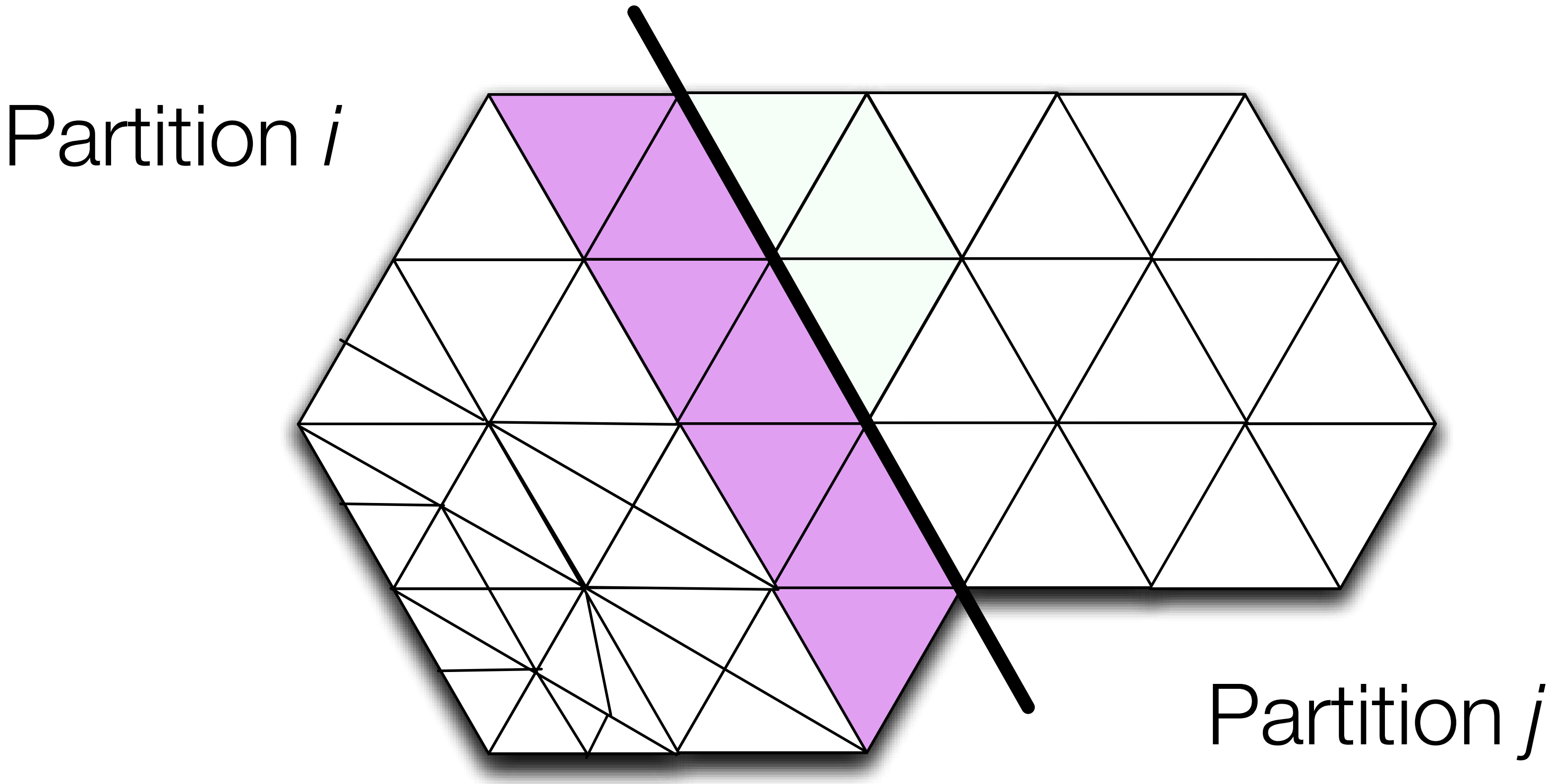
Initially, $\lambda_i \approx \lambda_j$ so no load balancing is needed.

Diffusion Load Balancing



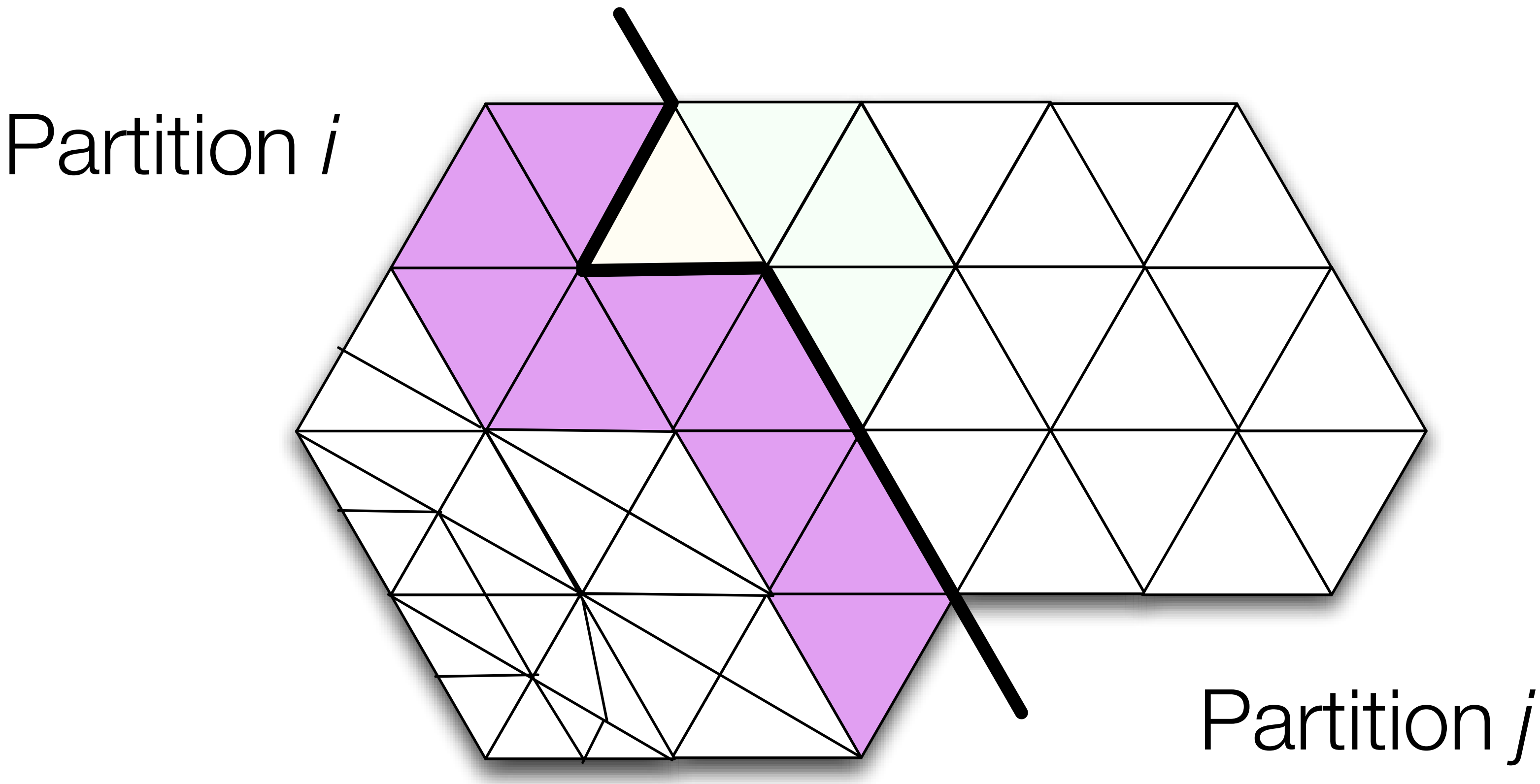
After local refinement, $\lambda_i > r\lambda_j$ so boundary elements will move from *i* to *j*

Diffusion Load Balancing



After local refinement, $\lambda_i > r\lambda_j$ so boundary elements will move from *i* to *j*

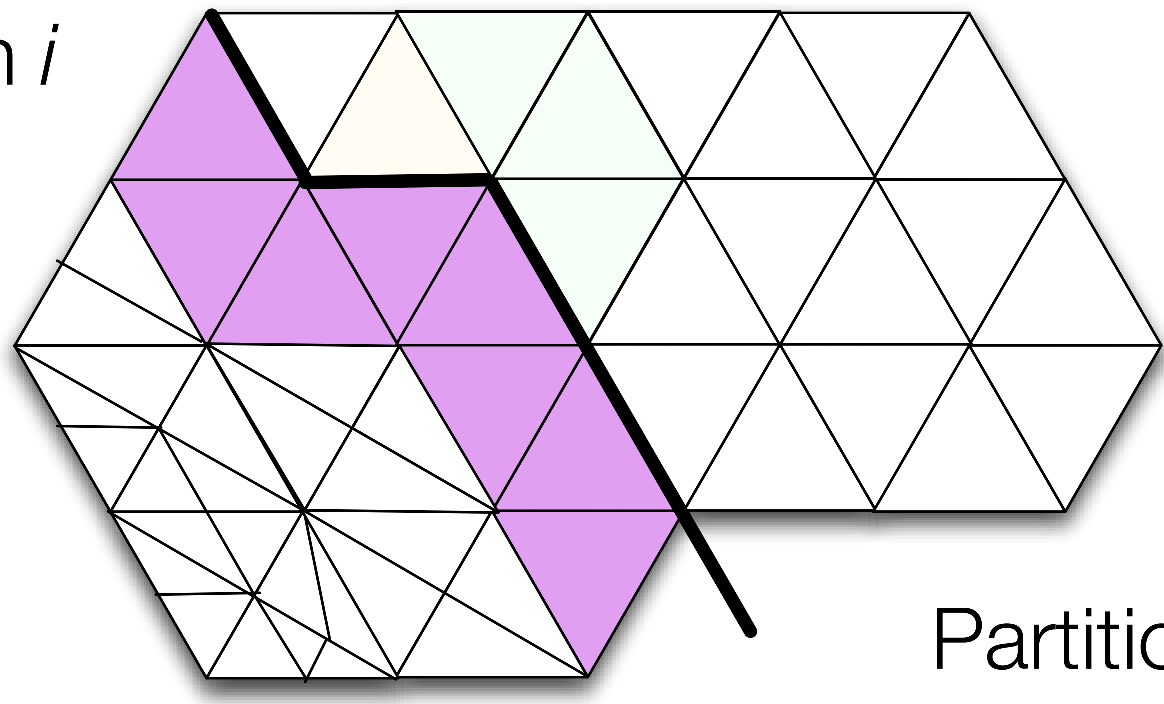
Diffusion Load Balancing



After local refinement, $\lambda_i > r\lambda_j$ so boundary elements will move from *i* to *j*

Diffusion Load Balancing

Partition i

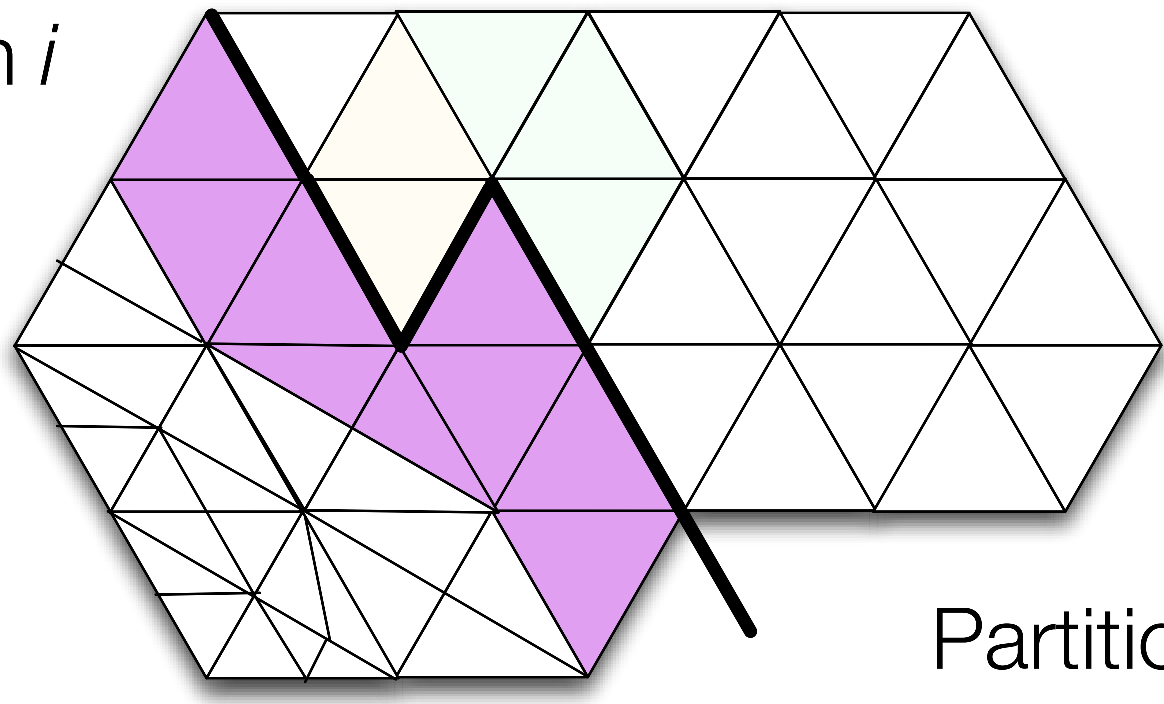


Partition j

After local refinement, $\lambda_i > r\lambda_j$ so boundary elements will move from i to j

Diffusion Load Balancing

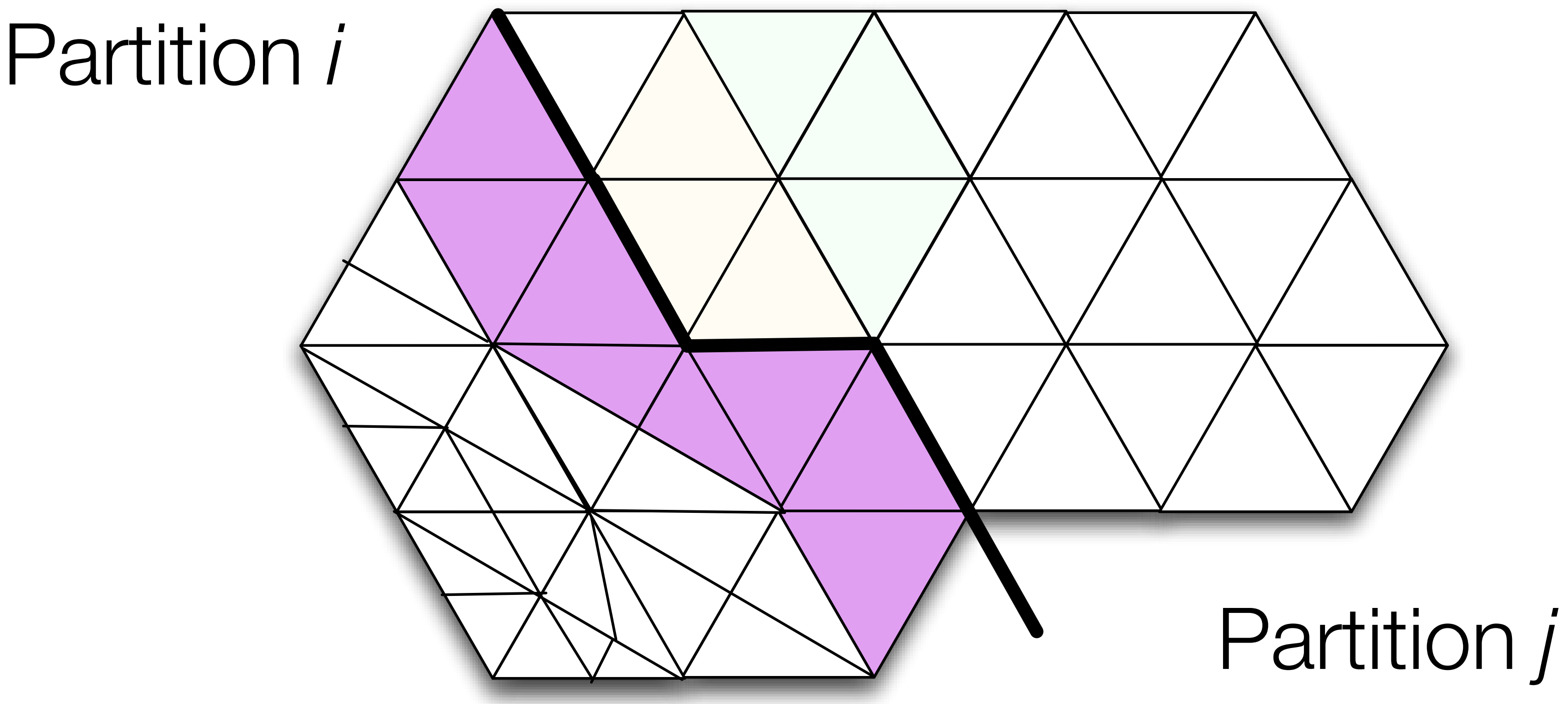
Partition *i*



Partition *j*

After local refinement, $\lambda_i > r\lambda_j$ so boundary elements will move from *i* to *j*

Diffusion Load Balancing



We attempt to migrate elements in a way that maintains or improves boundary quality.

Diffusion Load Balancing Issues

- Maintaining boundary quality
- Maintaining accurate load estimates
- Choosing *r* to avoid unneeded transfers while still avoiding serious imbalance
- Determining the right termination condition for the load balancing step
- Minimizing lock contention on boundary elements

Load Balancing Techniques for Asynchronous Spacetime Discontinuous Galerkin Methods

Aaron K. Becker (abecker3@illinois.edu)
Robert B. Haber
Laxmikant V. Kalé

University of Illinois, Urbana-Champaign

Parallel Programming Lab
Center for Process Simulation and Design

UNSCCM ’09