ParFUM:

A **Par**allel **F**ramework for **U**nstructured **M**eshes

Aaron Becker, Isaac Dooley, Terry Wilmarth, Sayantan Chakravorty

Charm++ Workshop 2008

What is ParFUM?

- A framework for writing parallel finite element codes
- Takes care of difficult tasks involved in parallelizing a serial code
- Provides advanced mesh operations such as mesh adaptivity and dynamic load balancing
- Constantly evolving to support application needs (for example, cohesive elements and collision detection)
- Based on Charm++ and AMPI. Supports C, C++, and Fortran

Making Parallel Finite Element Codes Easier

A simple **serial** finite element code:

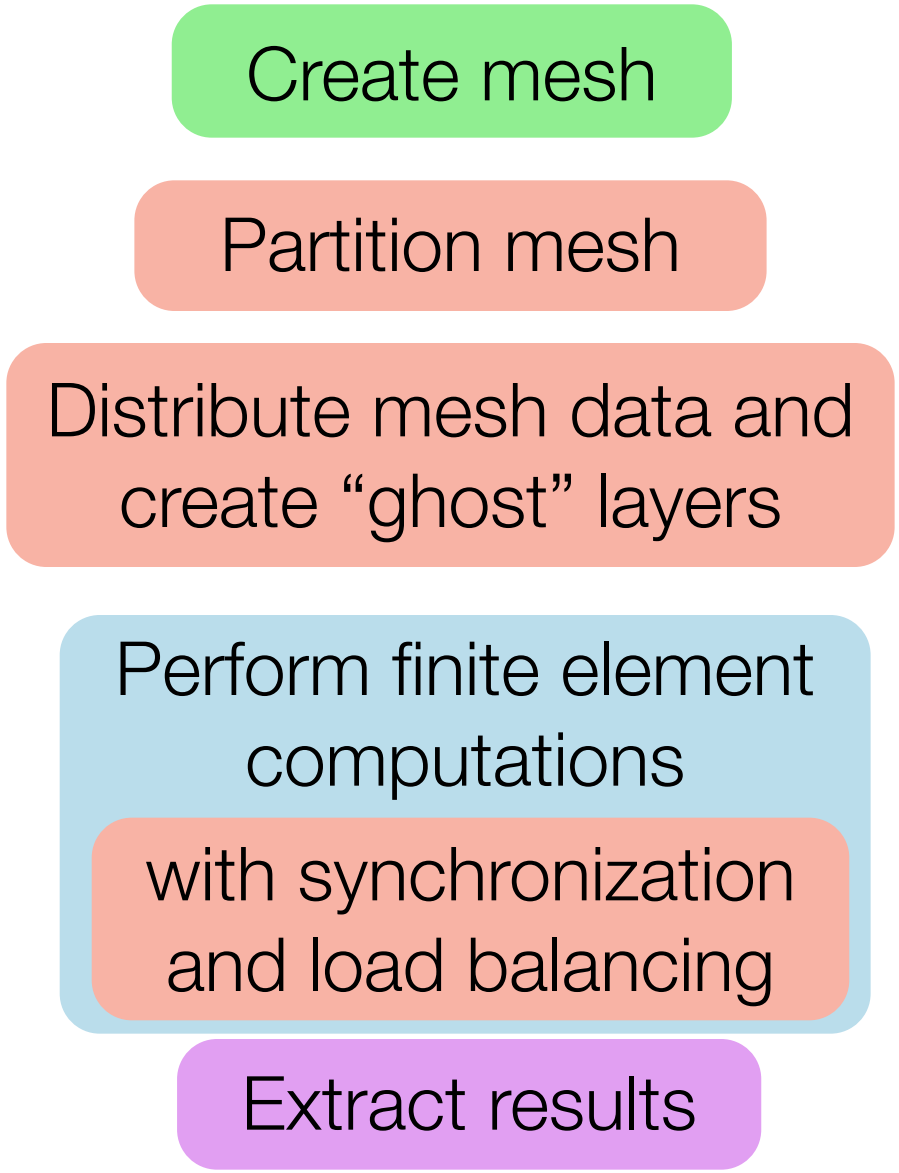
Create mesh

Perform finite element computations

Extract results

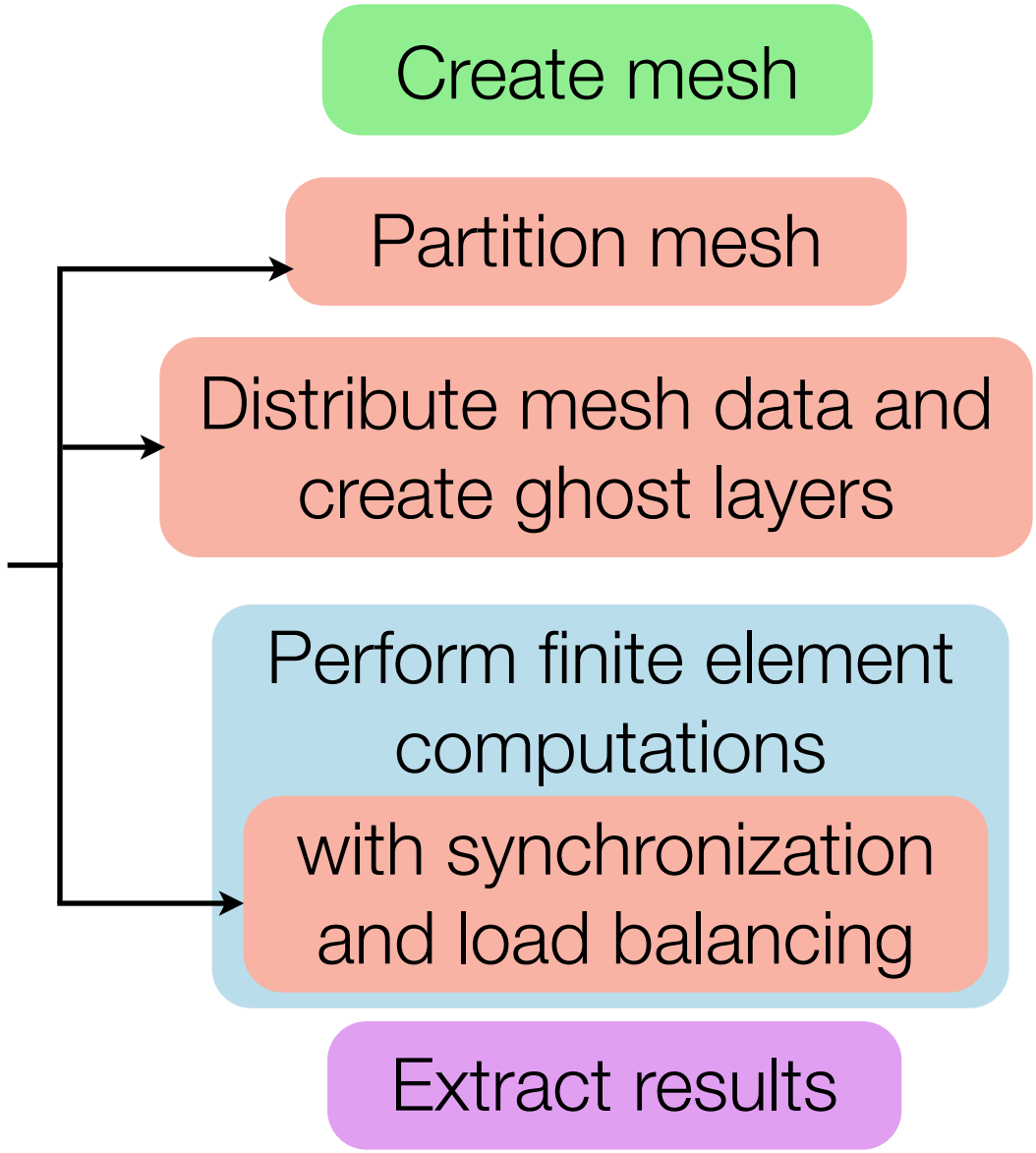
Making Parallel Finite Element Codes Easier

A simple **parallel** finite element code:

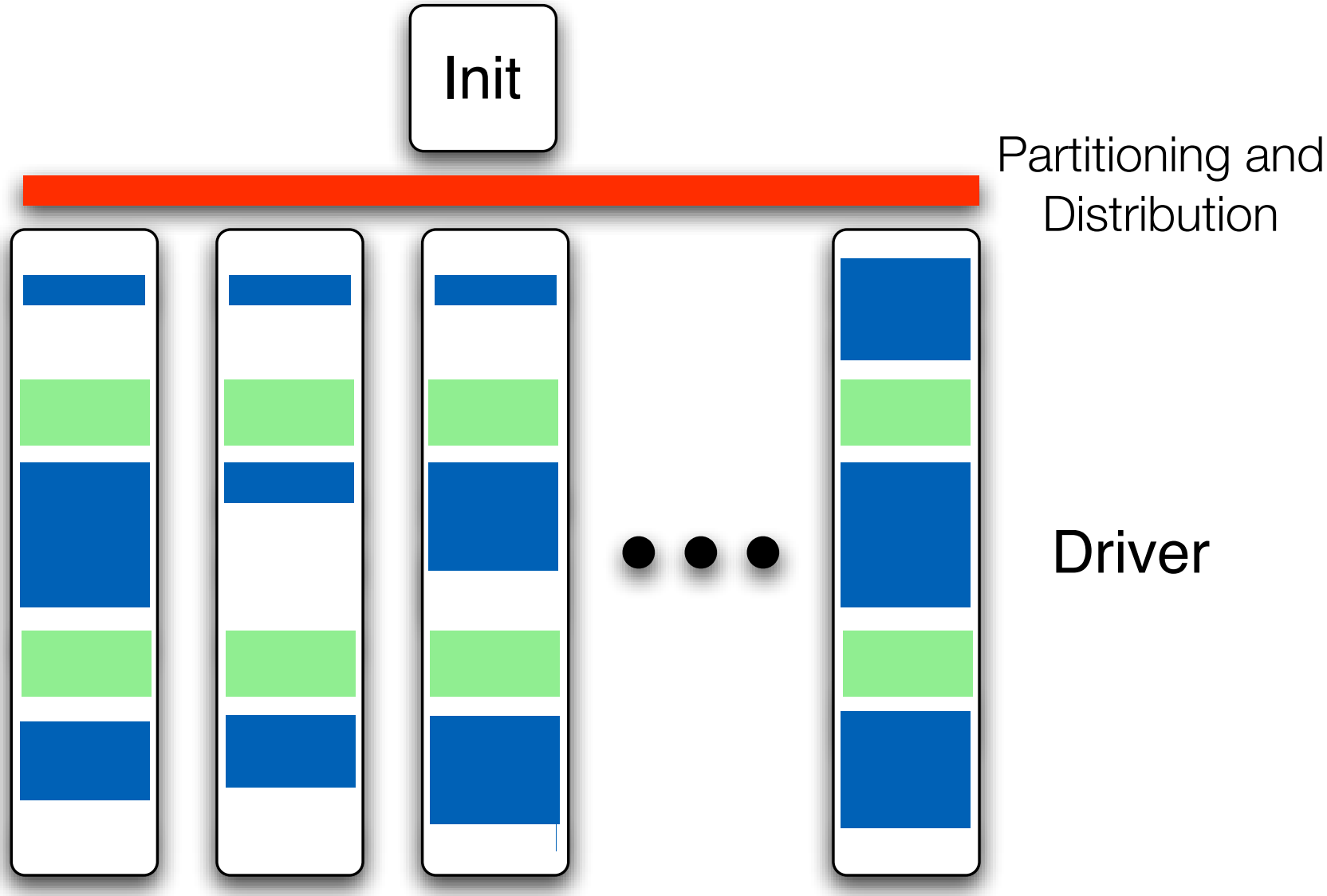


Making Parallel Finite Element Codes Easier

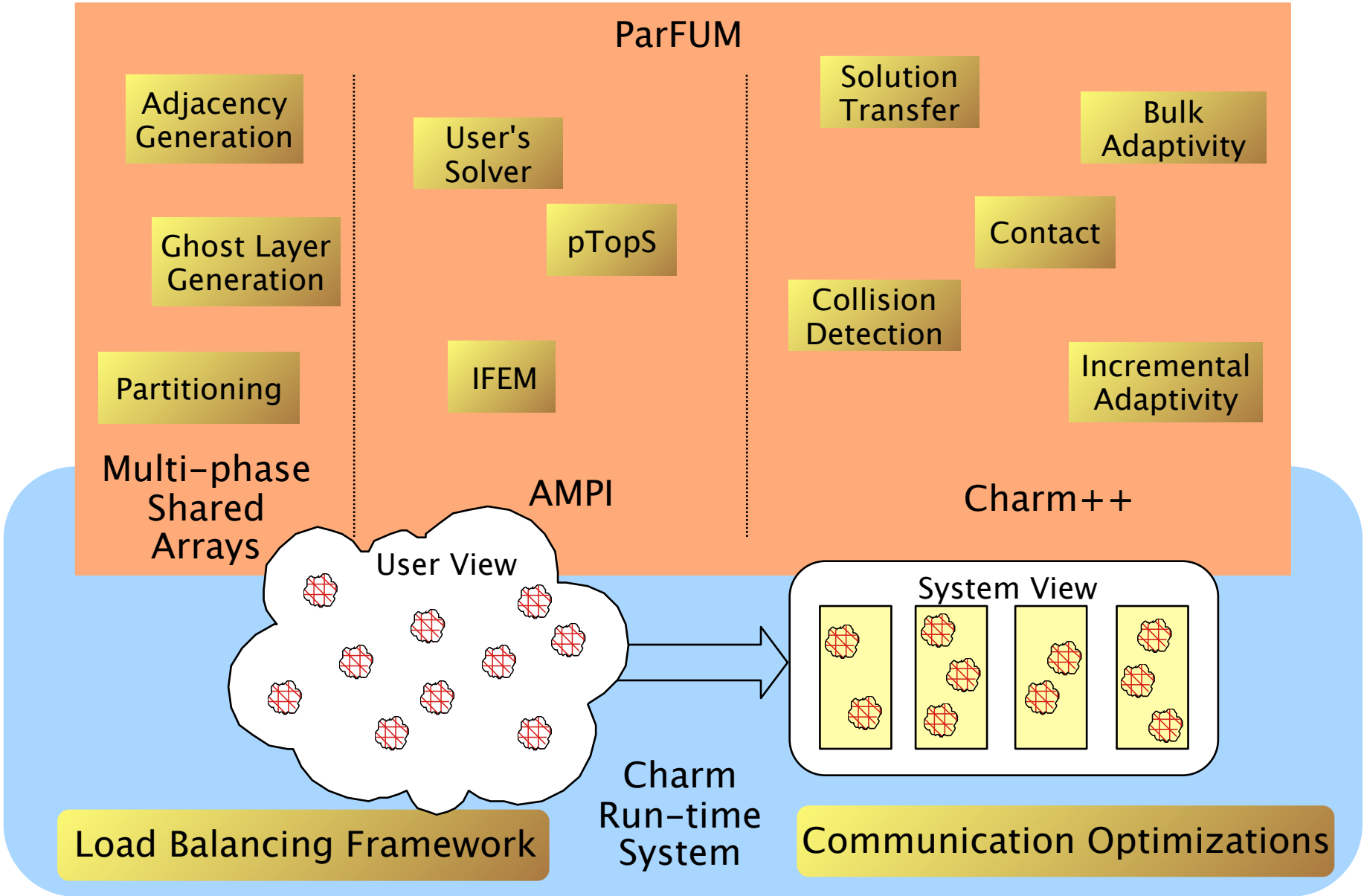
ParFUM can do these things automatically and let the developer concentrate on science and engineering



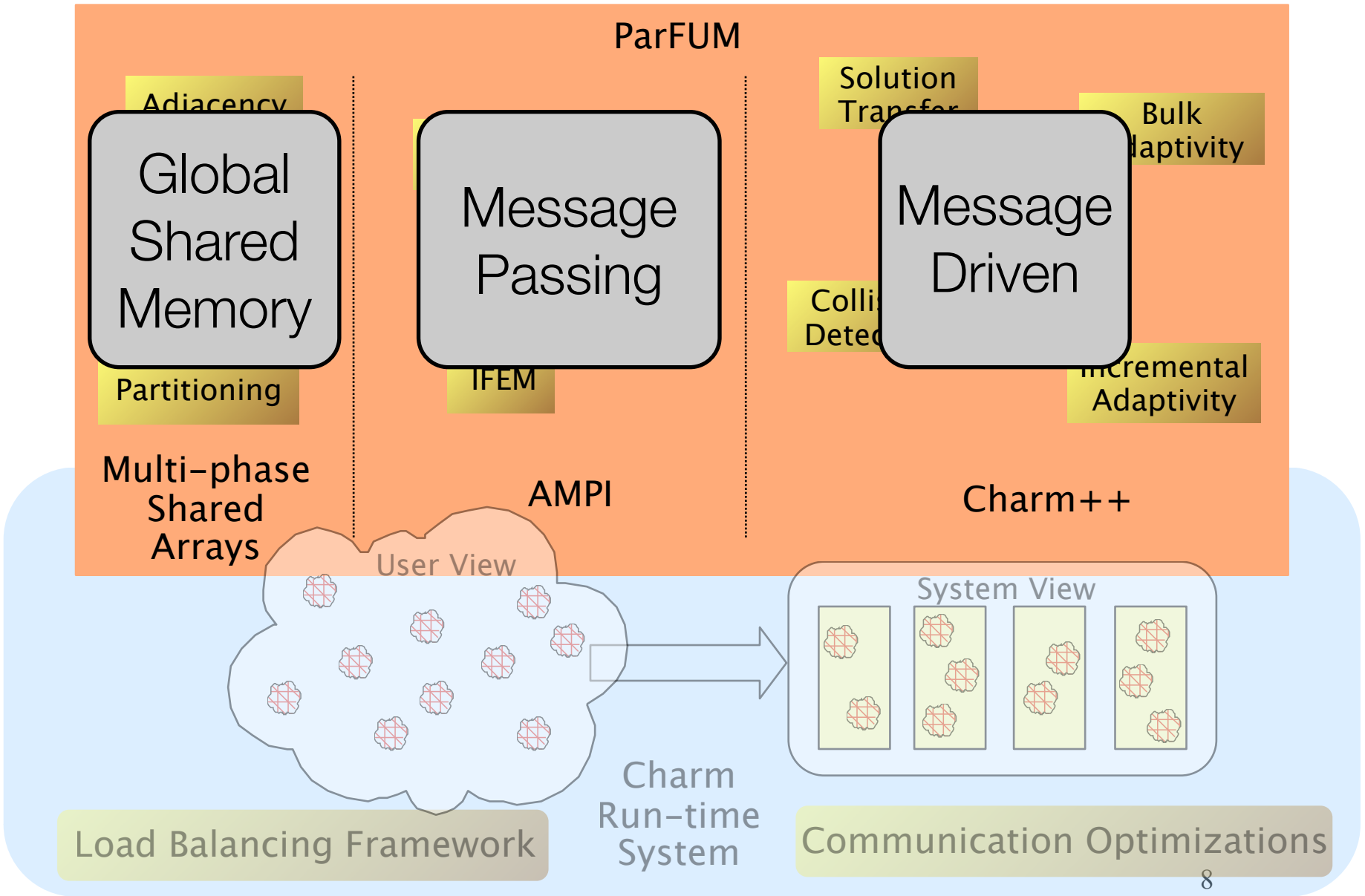
The Structure of a ParFUM Program



The Big Picture



Integrating Multiple Programming Models



ParFUM and AMPI

- Application code is written in AMPI, an implementation of MPI on top of the Charm RTS.
- AMPI processes (virtual processors, or VPs) are not tied to a physical processor, they can migrate and there may be many of them per physical processor
- This allows easier porting of MPI codes and eases the learning curve of ParFUM

Virtualization Tradeoffs

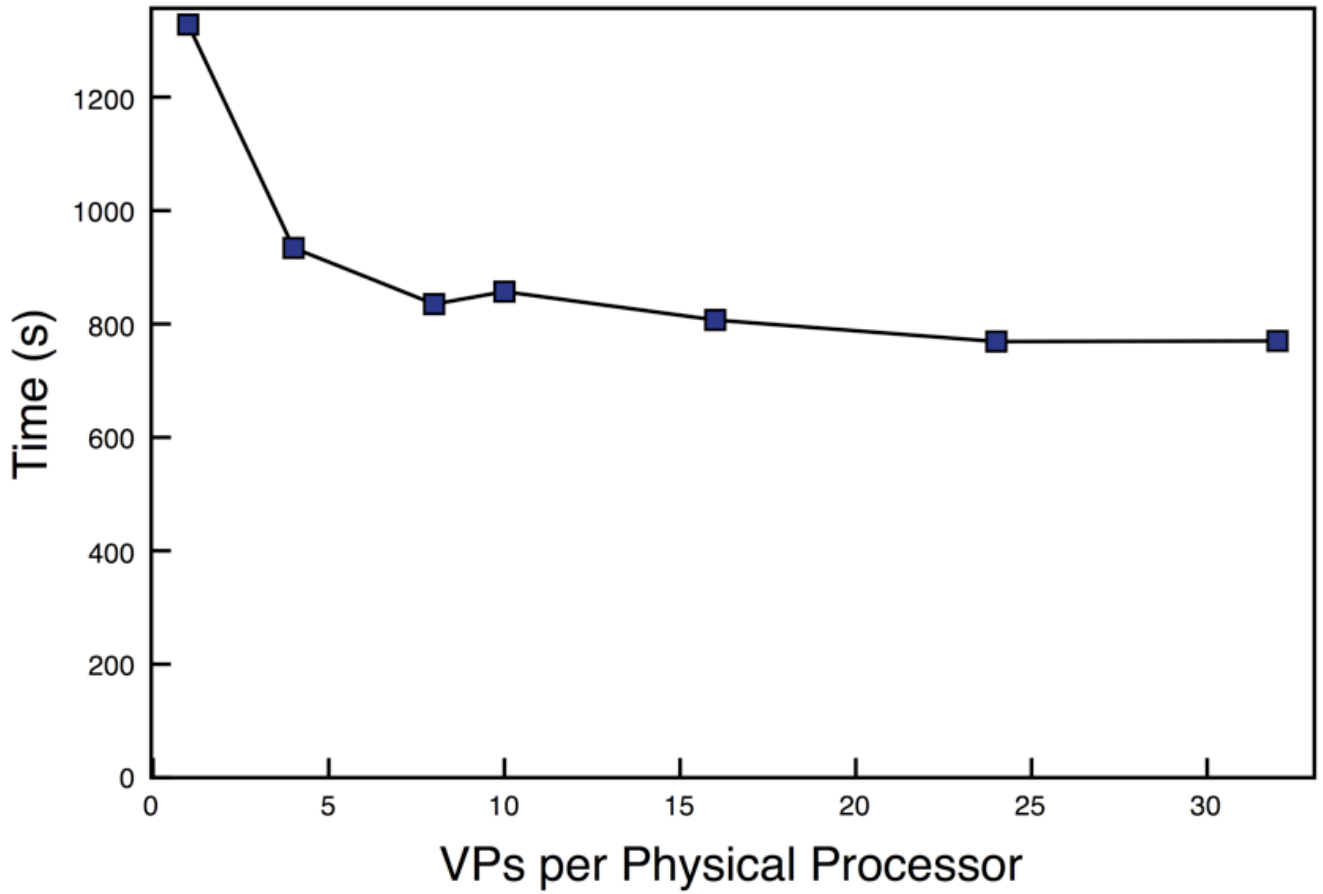
Advantages

- Allows adaptive overlap of communication and computation
- More granular load balancing
- Improved cache performance
- More flexibility

Disadvantages

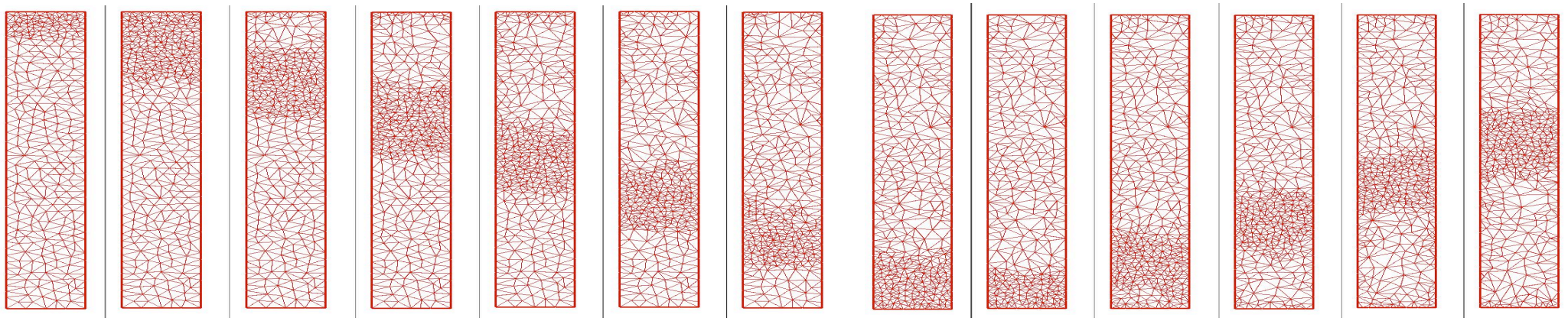
- More communication
- Worse ratio of remote data to local data
- Imposes some thread overhead
- High virtualization requires many elements per node

Virtualization Performance Impact



For this dynamic fracture code, virtualization provides a substantial benefit

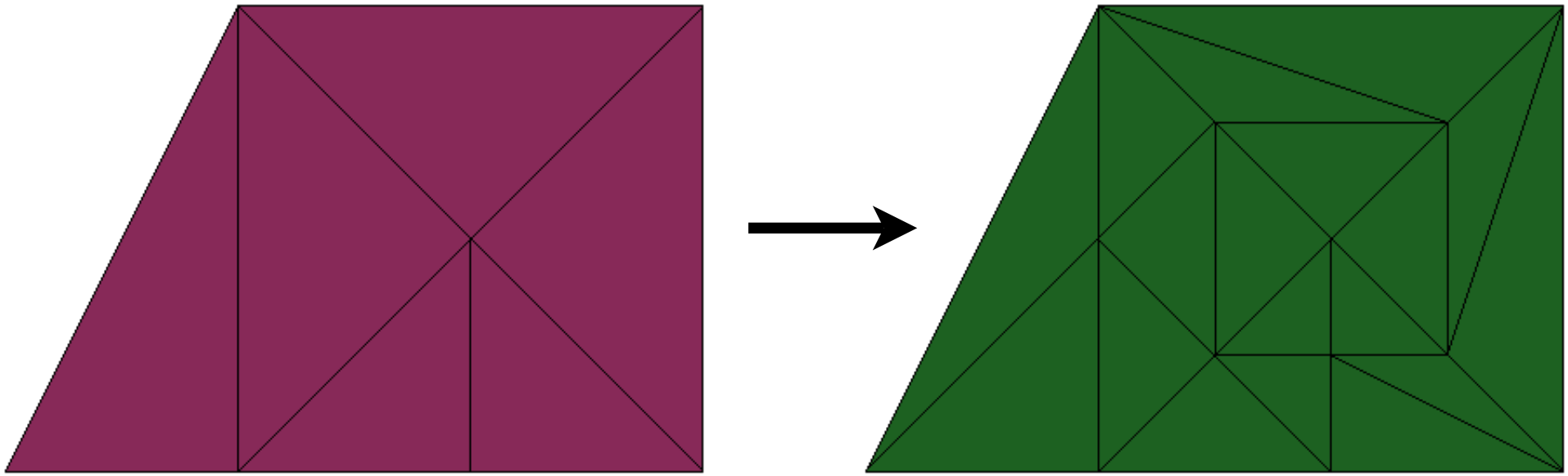
Parallel Mesh Adaptivity



- Efficient parallel adaptivity is critical for many unstructured mesh codes
- ParFUM provides two implementations of common operations:
  - incremental (2D triangle meshes): each individual operation leaves the mesh consistent. Relatively slow and puts limitations on ghost layers
  - bulk (2D triangles and 3D tets): many operations performed at once, ghosts and adjacencies updated at end. Lower cost, no restrictions on ghost layers. (ongoing work)

Higher Level Adaptivity

Propagating Edge Bisection

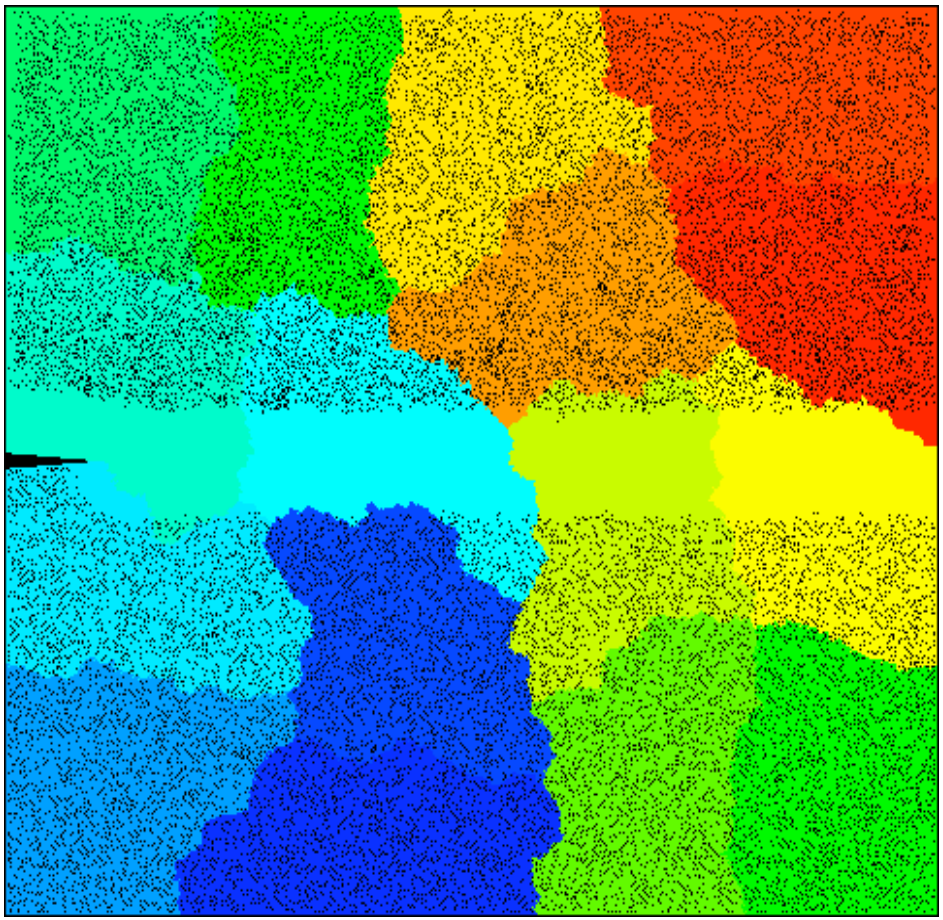


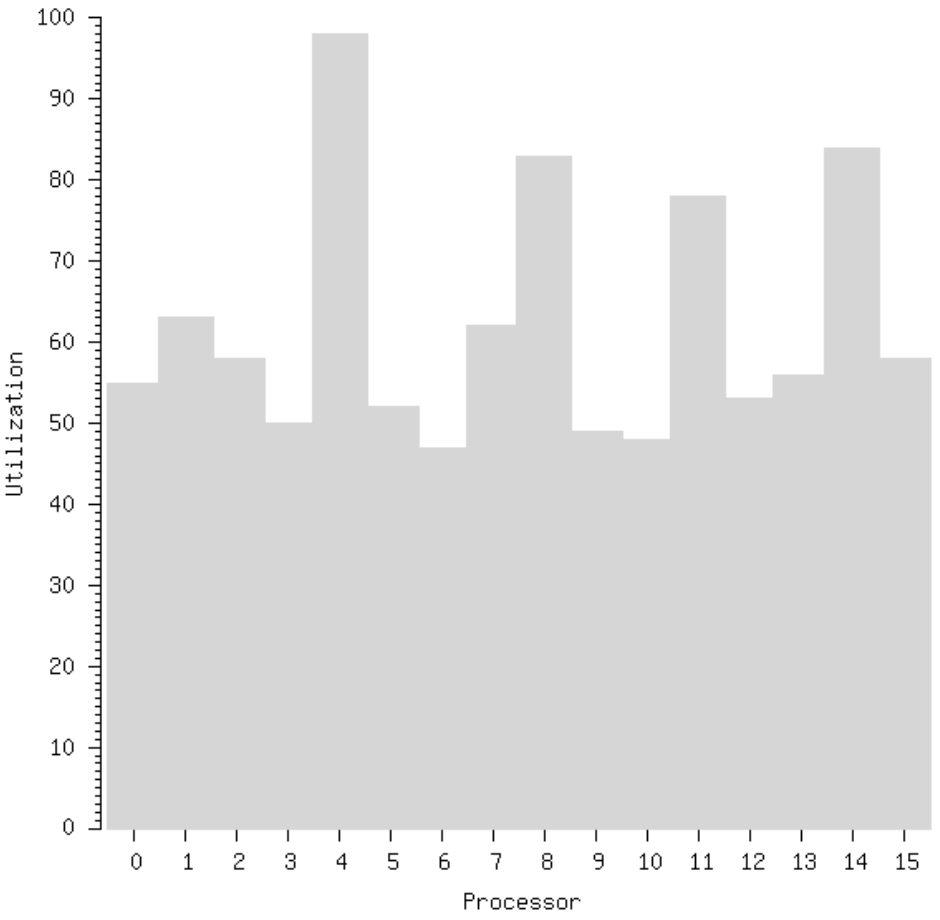
- Operations like propagating edge bisection are composed from edge bisect, flip, and contraction primitives

- Which is better, bulk or incremental? Depends on amount and frequency of adaptivity

Load Balance, Adaptivity, and Virtualization

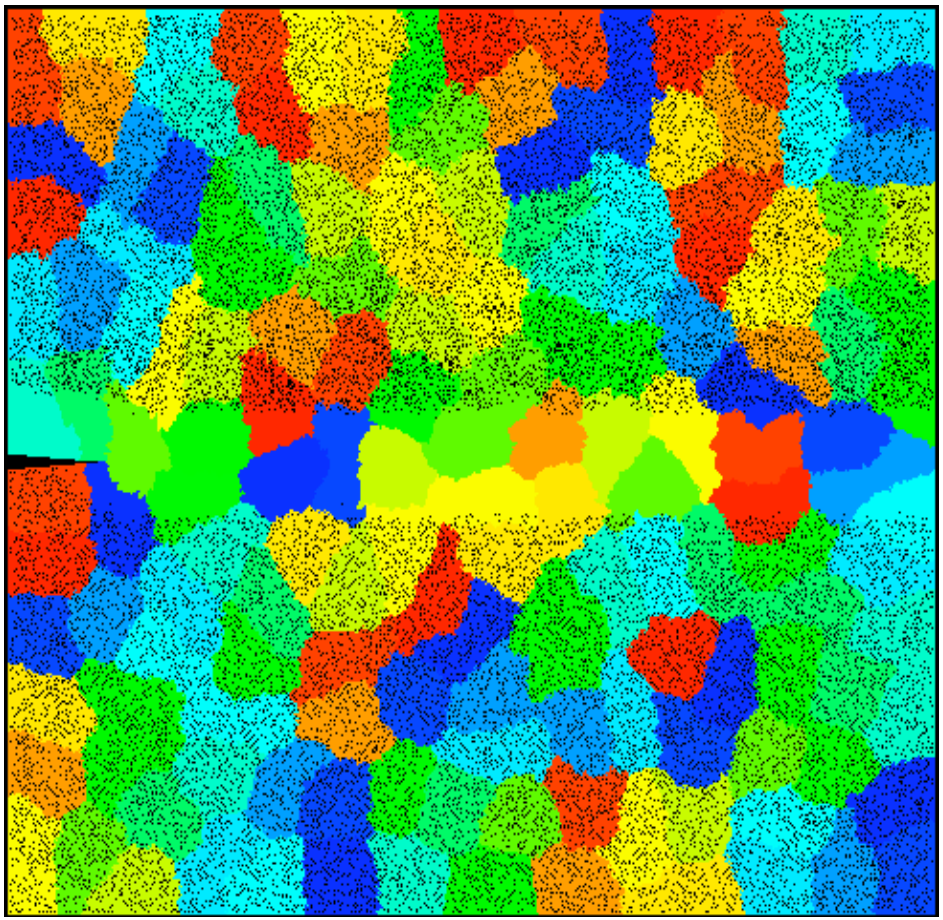
Serious load imbalance: areas near fracture are much more expensive

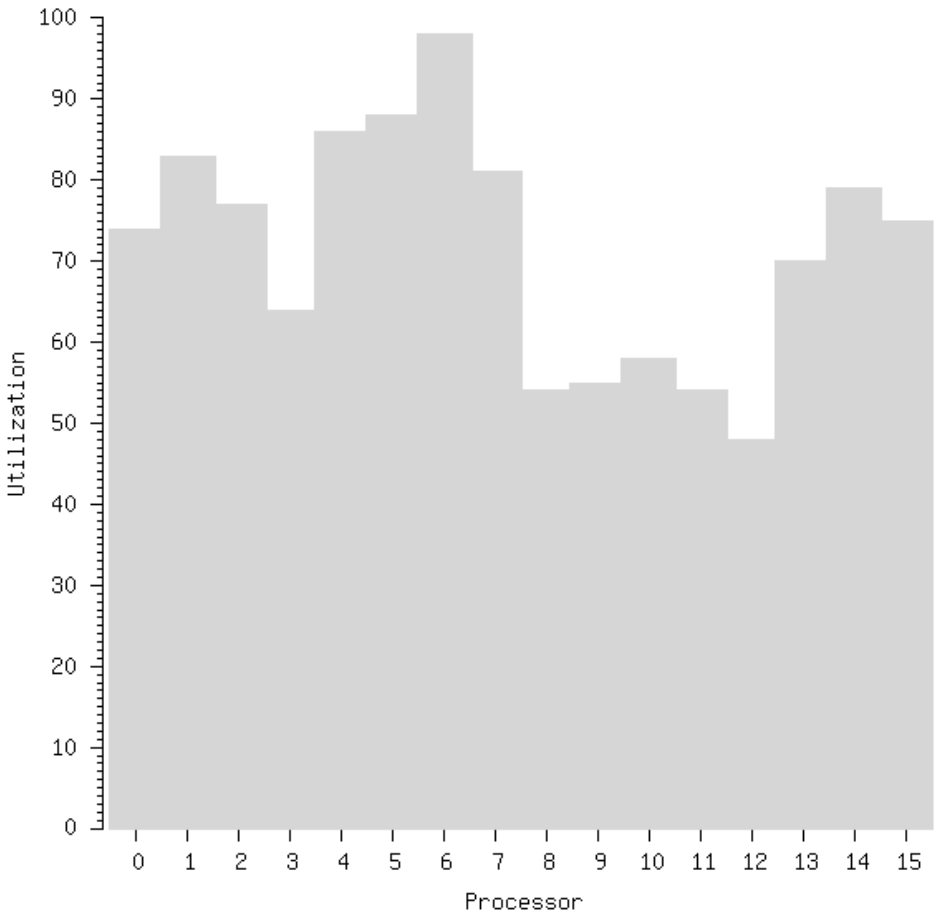




Load Balance, Adaptivity, and Virtualization

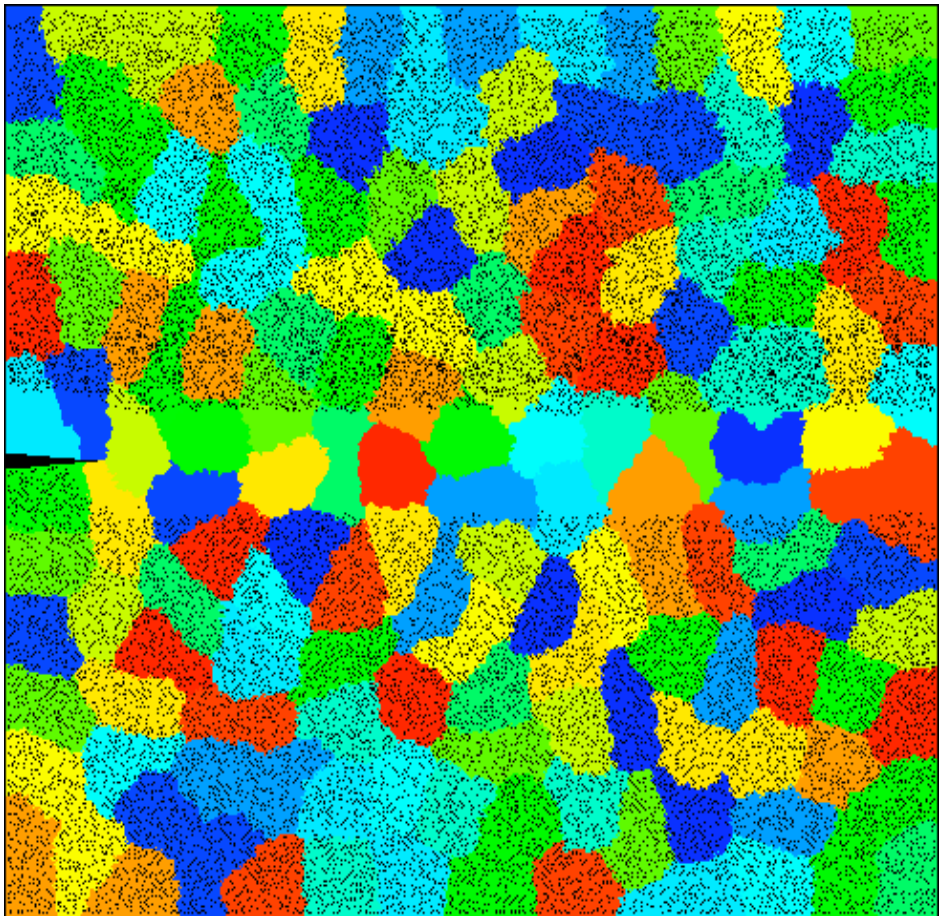
We can change the VP mapping to distribute computationally expensive parts of the mesh better

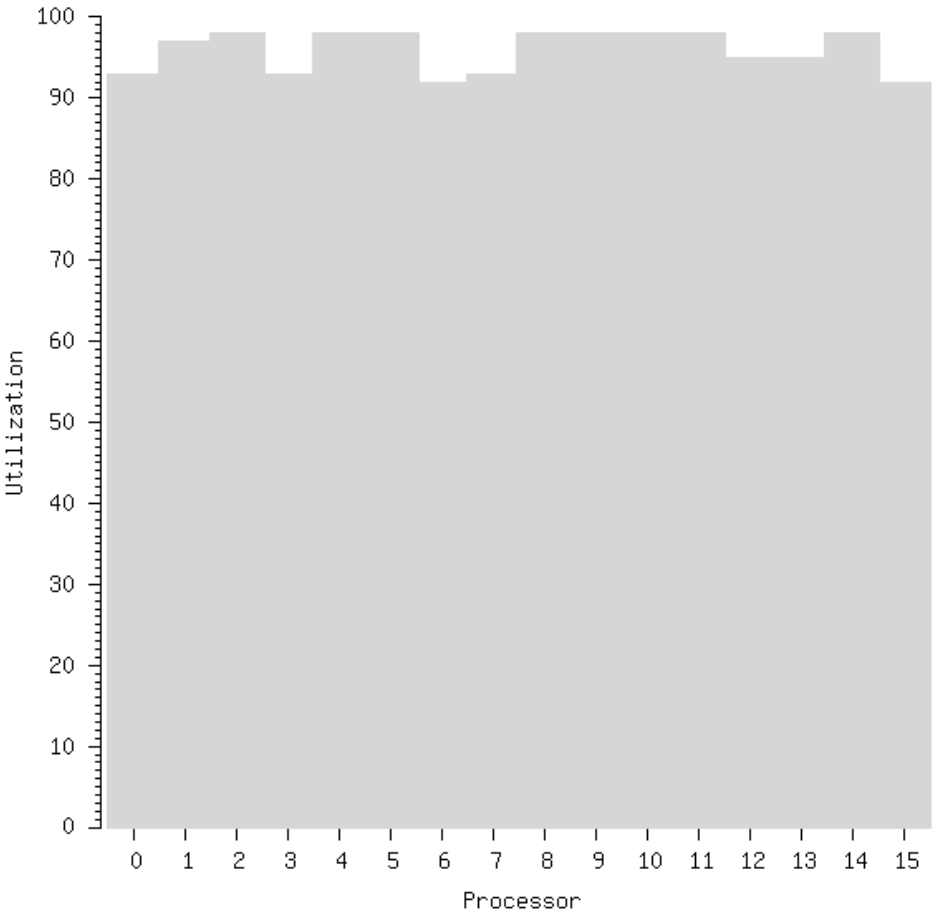




Load Balance, Adaptivity, and Virtualization

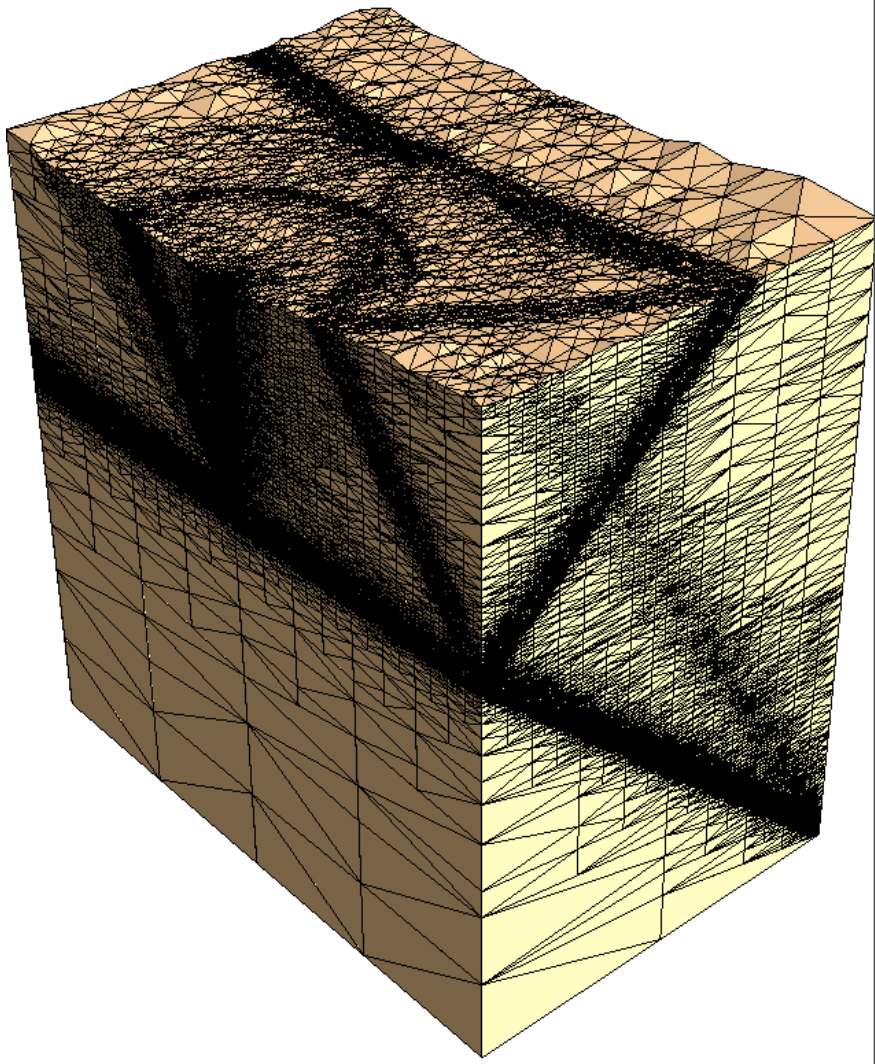
Assigning VPs using a greedy load balancer further improves utilization





Spacetime Meshing

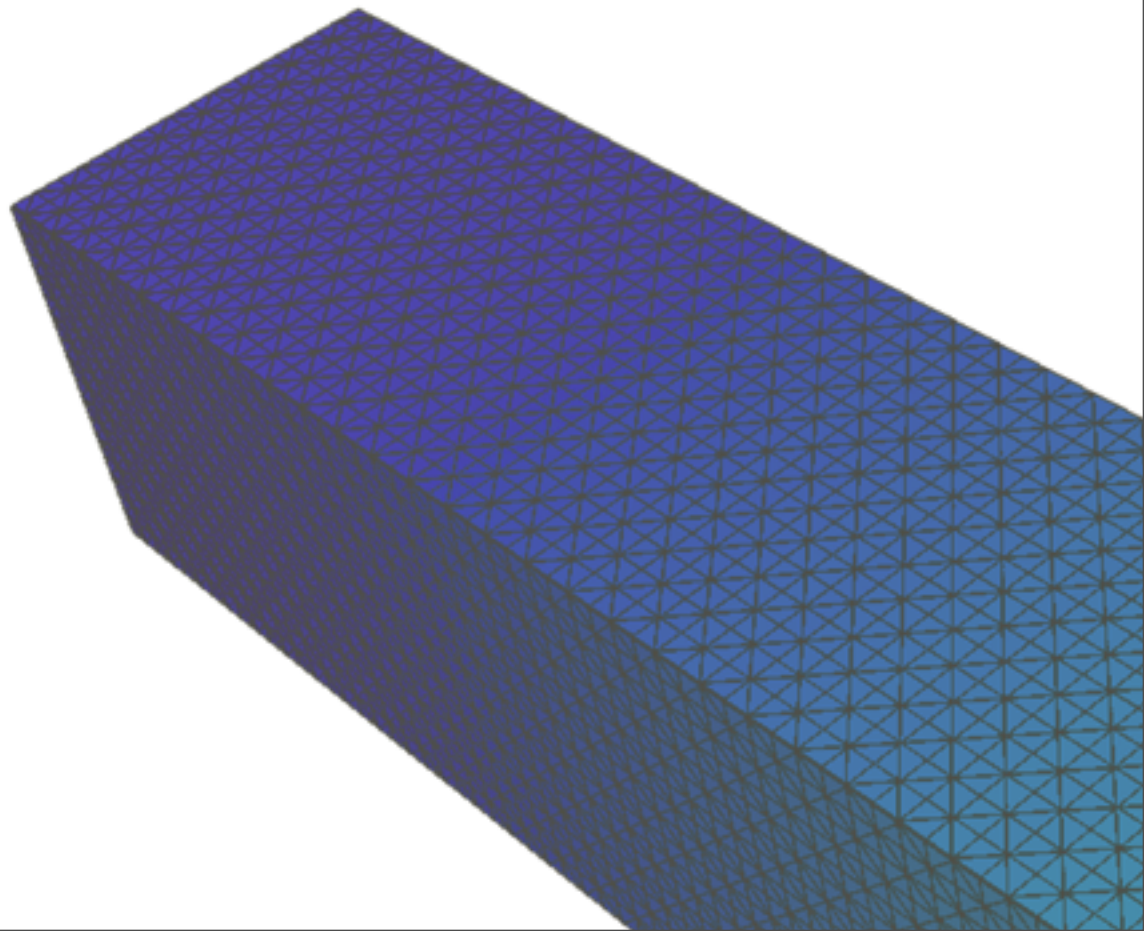
- Parallelization of Spacetime Discontinuous Galerkin (SDG) algorithm [Haber]
- Adaptive in both space and time, uses incremental adaptivity
- Asynchronous code, no global barriers



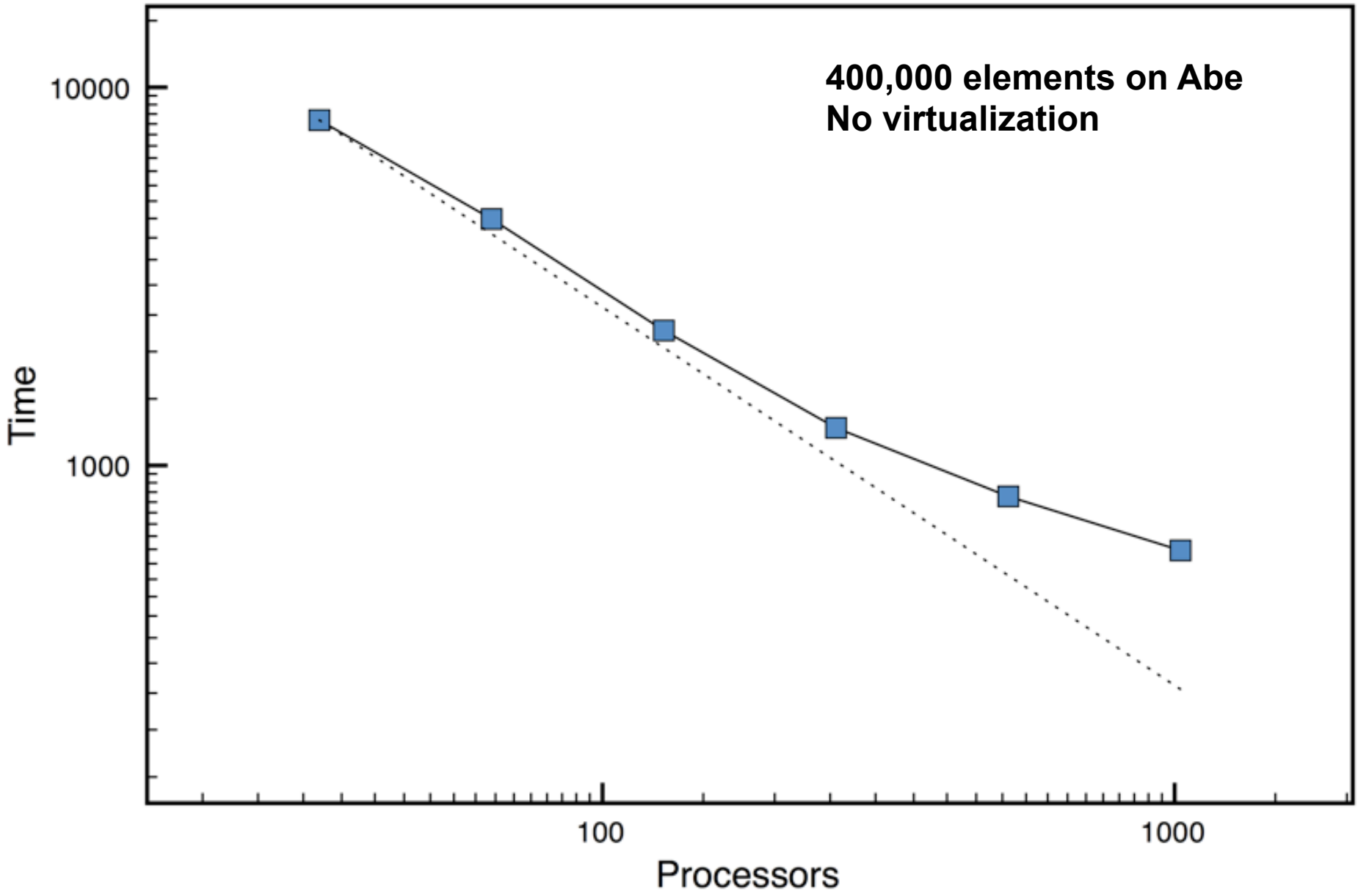
PTops

- Structural dynamics code for graded materials [Paulino]

- Based on Tops, a serial framework featuring an efficient topological mesh representation



PTops Strong Scaling

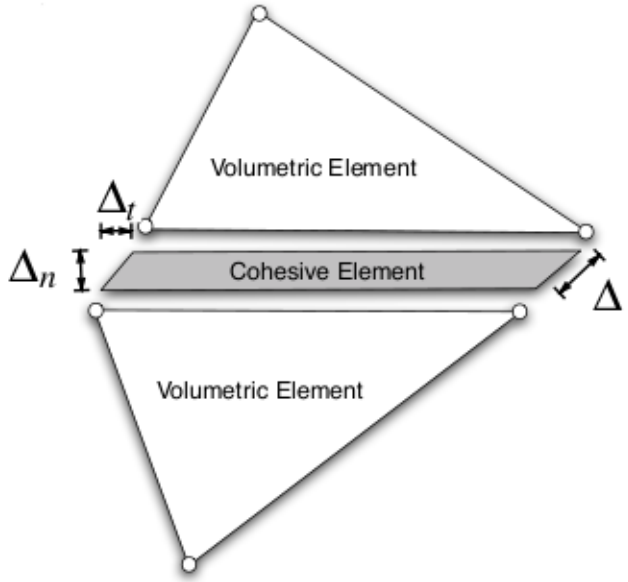


PTops and CUDA

- ParFUM-Tops interface has CUDA support
- Our implementation runs ~10x faster on a single node using CUDA
- Limited usefulness due to lack of double precision and lack of access to clusters which combine GPUs and high quality interconnects

Ongoing Work

- On-demand insertion of cohesive elements (truly extrinsic cohesives) in PTops for dynamic fracture simulations
- Efficient, scalable implementation of bulk edge flip and edge contraction
- Contact: use Charm++ collision detection library to detect when domain fragments come into contact



ParFUM:

A **Par**allel **F**ramework for **U**nstructured **M**eshes

Aaron Becker, Sayantan Chakravorty, Isaac Dooley, Terry Wilmarth
Parallel Programming Lab
Charm++ Workshop 2008