CRAY
THE SUPERCOMPUTER COMPANY

CHAPEL LULESH

Episodes IV, V, and VI

SC12: November 14th, 2012

Sung-Eun Choi

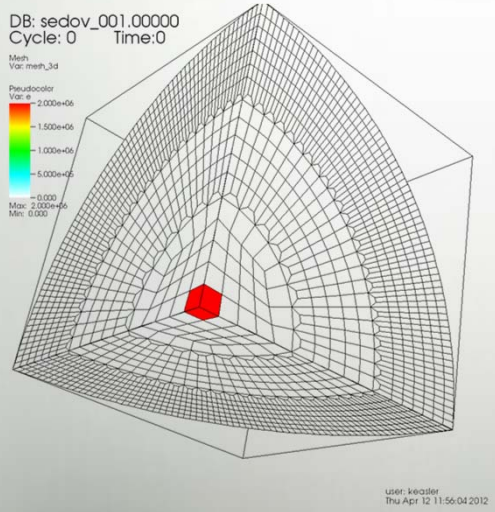
Chapel Team, Cray Inc.

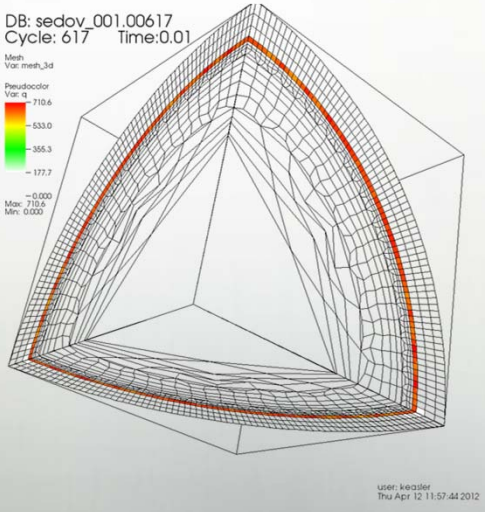
SC12
Salt Lake City, Utah

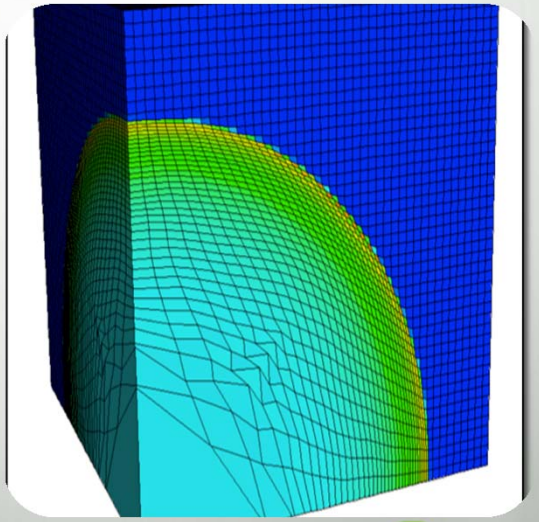
LULESH

LULESH:

- Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics challenge problem
- developed by LLNL under DARPA UHPC
- serves as a proxy app for key computation patterns
- https://computation.llnl.gov/casc/ShockHydro/

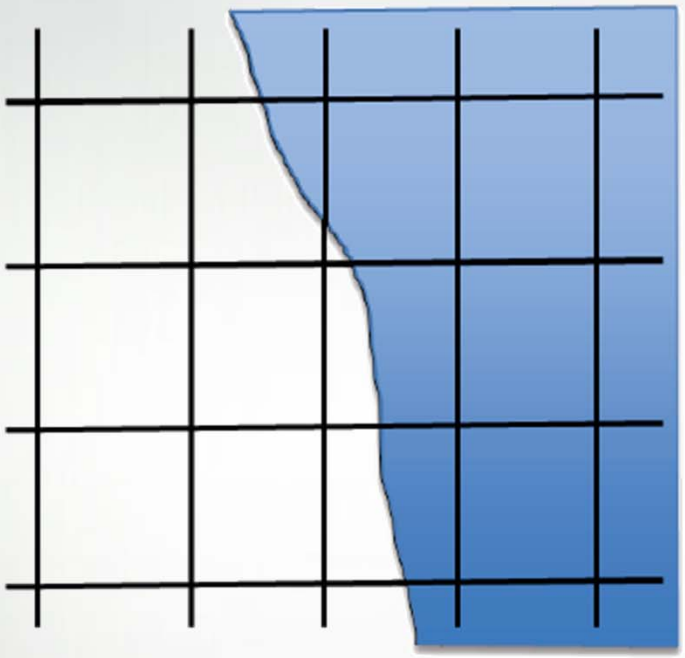




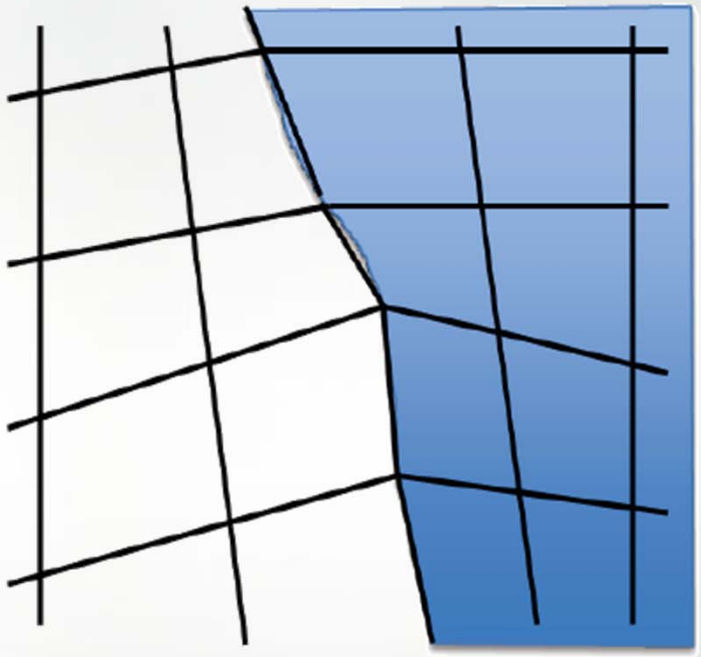


pictures courtesy of Rob Neely, Bert Still, Jeff Keasler, LLNL

Eulerian vs. Lagrangian Meshes



Eulerian mesh
(grid stays fixed)



Lagrangian mesh
(grid adapts to materials)

Image Source: LULESH specification, LLNL-TR-490254
https://computation.llnl.gov/casc/ShockHydro/

Episode IV

A New Hope

SC12
Salt Lake City, Utah

LULESH Co-Design History: Cray/LLNL

Apr 2011: LLNL expresses interest in Chapel at Salishan

- made us aware of the LULESH benchmark

Summer 2011: Cray intern ports LULESH to Chapel

- *caveat:* used structured mesh to represent data arrays

Episode V

Chapel Strikes Back

SC12
Salt Lake City, Utah

LULESH Co-Design History: Cray/LLNL

Apr 2011: LLNL expresses interest in Chapel at Salishan

- made us aware of the LULESH benchmark

Summer 2011: Cray intern ports LULESH to Chapel

- *caveat:* used structured mesh to represent data arrays

Nov 2011: Chapel team tunes LULESH for single-node performance

Dec 2011: Chapel team visits LLNL (talk, tutorial, 1-on-1 sessions)

Mar 2012: Jeff Keasler (LLNL) visits Cray to pair-program

- in one afternoon, converted from structured to unstructured mesh
- impact on code minimal (mostly in declarations) due to:
  - domains/arrays/iterators
  - rank-independent features

Episode VI

Return of the Salishan

SC12
Salt Lake City, Utah

LULESH Co-Design History: Cray/LLNL

Apr 2011: LLNL expresses interest in Chapel at Salishan
- made us aware of the LULESH benchmark

Summer 2011: Cray intern ports LULESH to Chapel
- *caveat:* used structured mesh to represent data arrays

Nov 2011: Chapel team tunes LULESH for single-node performance

Dec 2011: Chapel team visits LLNL (talk, tutorial, 1-on-1 sessions)

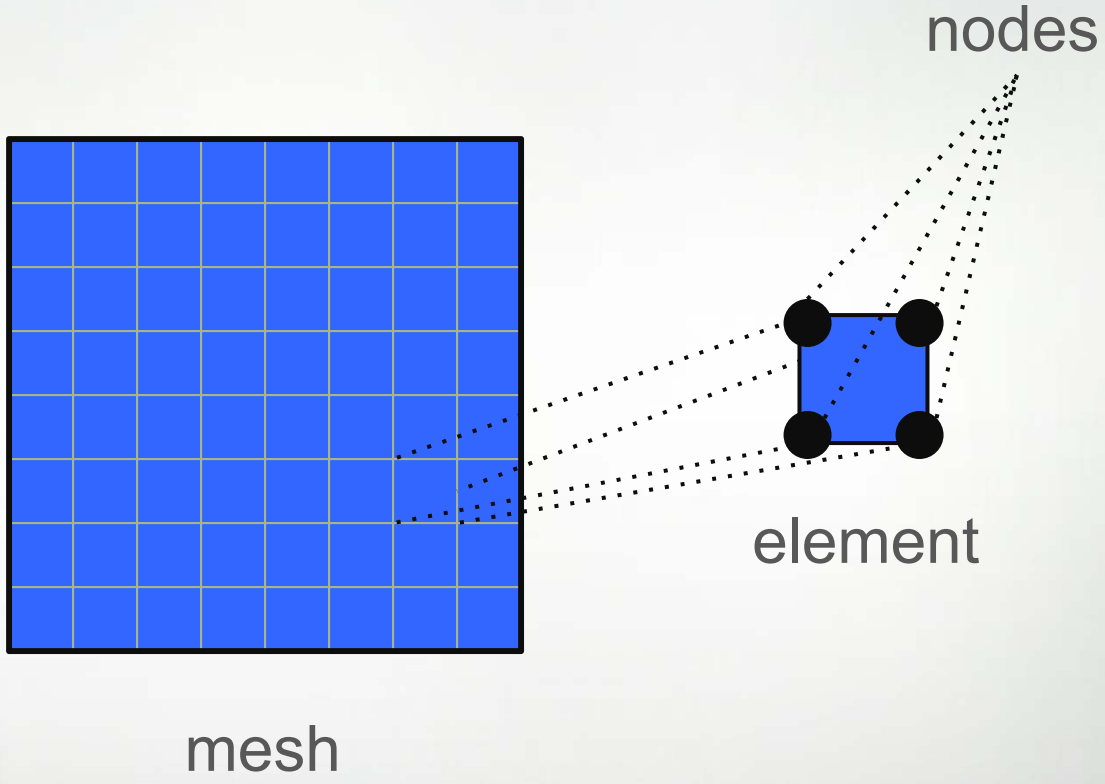
Mar 2012: Jeff Keasler (LLNL) visits Cray to pair-program
- in one afternoon, converted from structured to unstructured mesh
- impact on code minimal (mostly in declarations) due to:
  - domains/arrays/iterators
  - rank-independent features

Apr 2012: LLNL reports on collaboration at Salishan

Apr 2012: Chapel 1.5.0 release includes LULESH as an example code

Sep-Nov 2012: performance tuning, initial distributed sparse domains

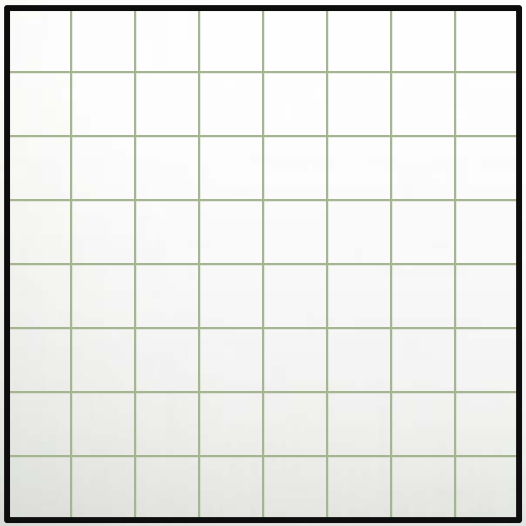
Fundamental LULESH concepts/terminology



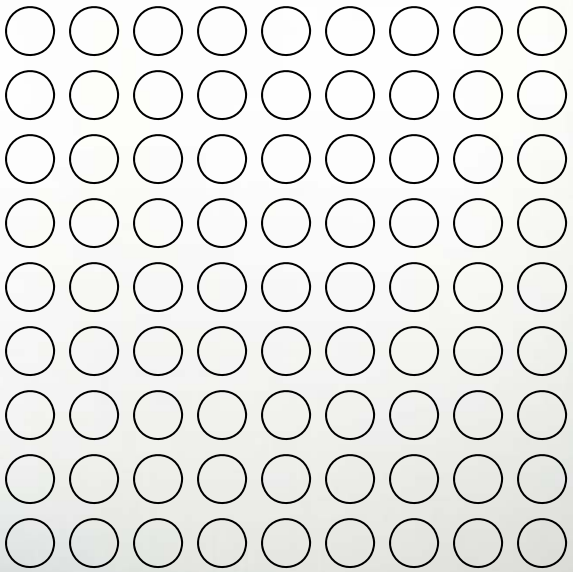
Representation of Concepts in Chapel

- Abstract Element and Node *Domains* (Views):

```
const nodesPerEdge = elemsPerEdge+1;
const ElemSpace = {0..#elemsPerEdge, 0..#elemsPerEdge},
      NodeSpace = {0..#nodesPerEdge, 0..#nodesPerEdge};
```



ElemSpace

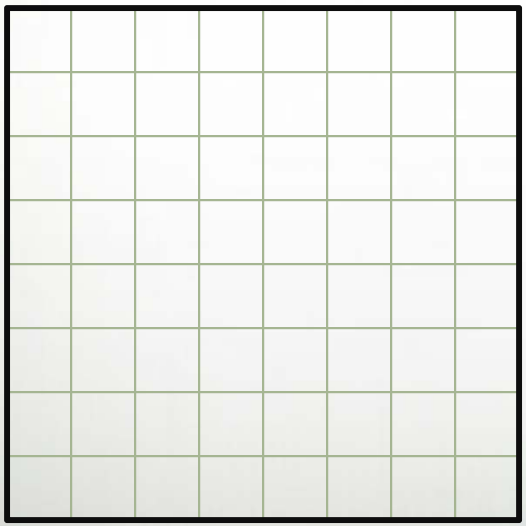


NodeSpace

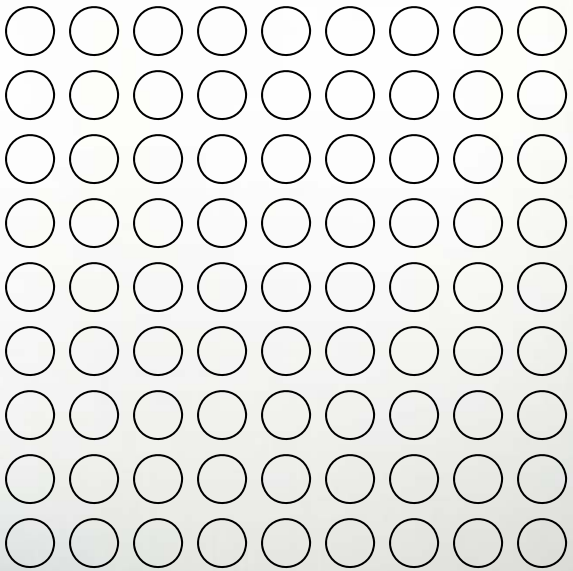
Representation of Concepts in Chapel

- Abstract Element and Node *Domains* (Views):

```
const ElemSpace = {0..#numElems},
      NodeSpace = {0..#numNodes};
```



ElemSpace

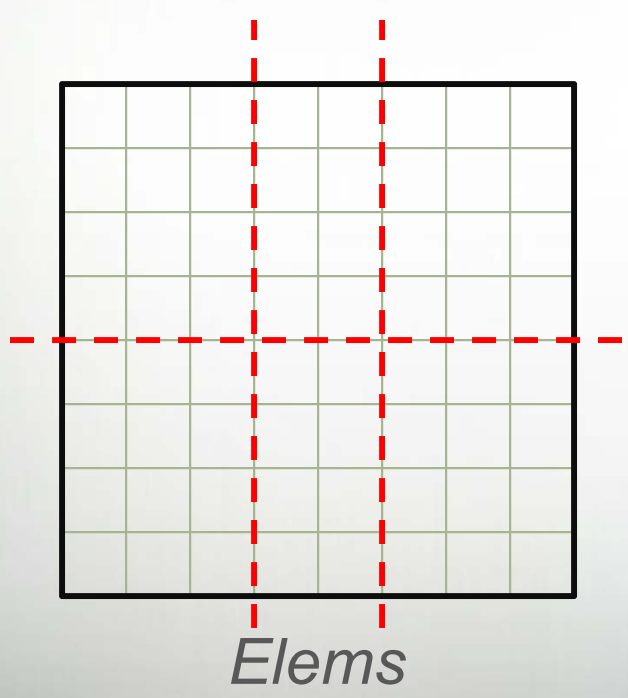


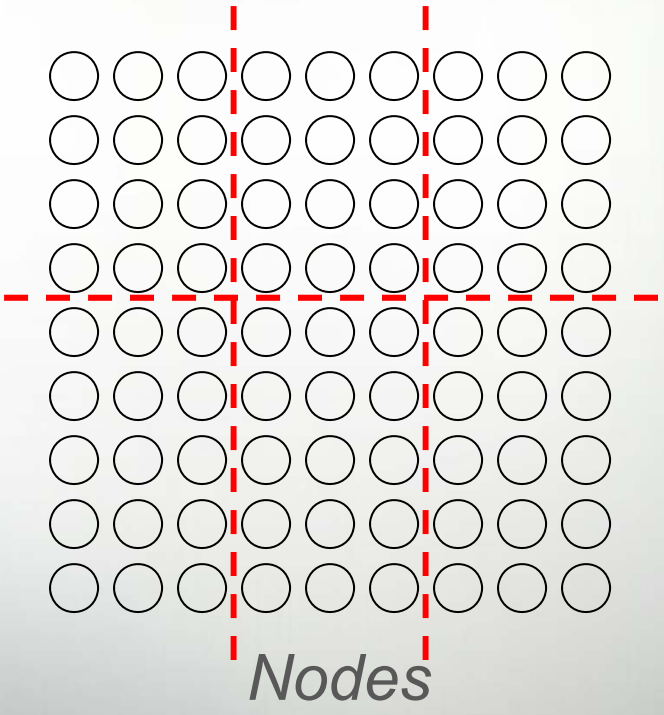
NodeSpace

Representation of Concepts in Chapel

- Distributed Element/Node Domains:

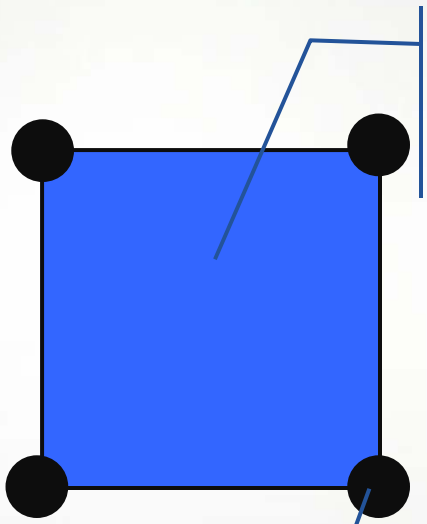
```
const Elems = ElemSpace dmapped Block(ElemSpace),
      Nodes = NodeSpace dmapped Block(NodeSpace);
```





Element vs. Node Fields

- Some variables (*fields*) are associated with elements, others with nodes.



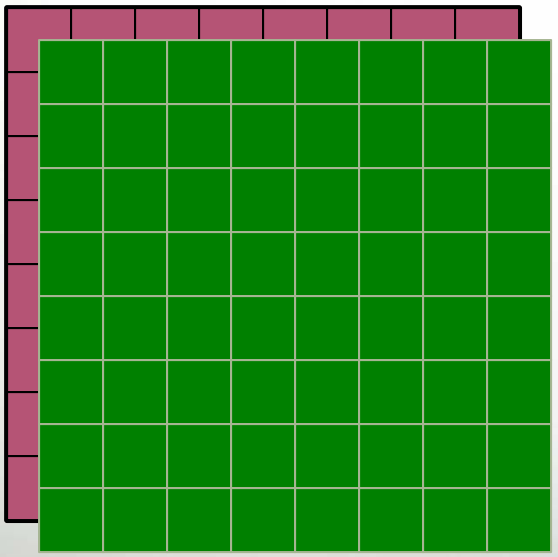
Pressure, energy, viscosity, volume, …

Position, velocity, acceleration, force, mass

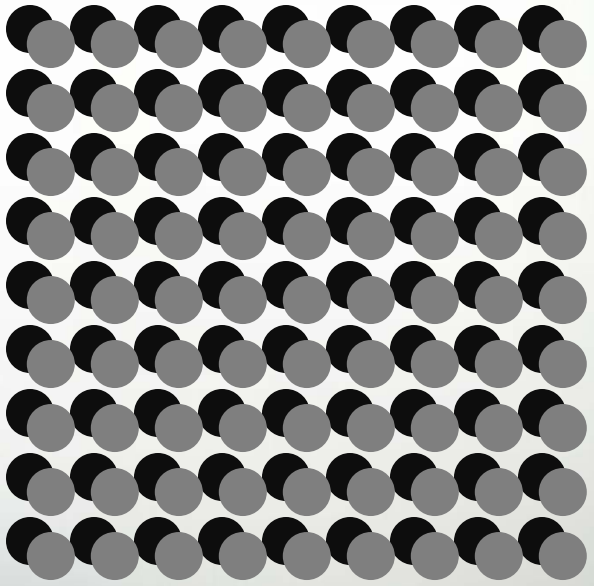
Representation of Fields in Chapel

- Sample field declarations:

```
var x, y, z: [Nodes] real;
var e, p: [Elems] real;
```



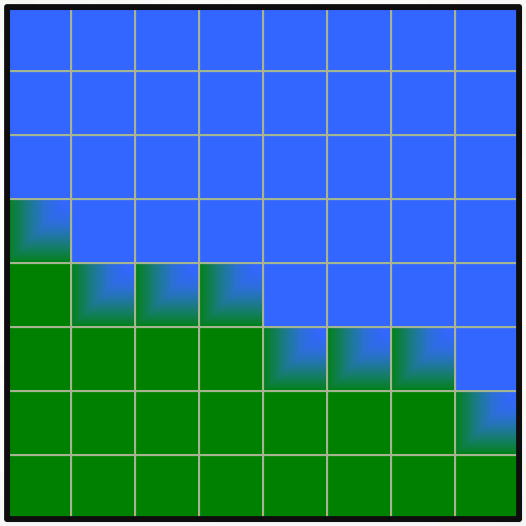
e, p



x, y, z

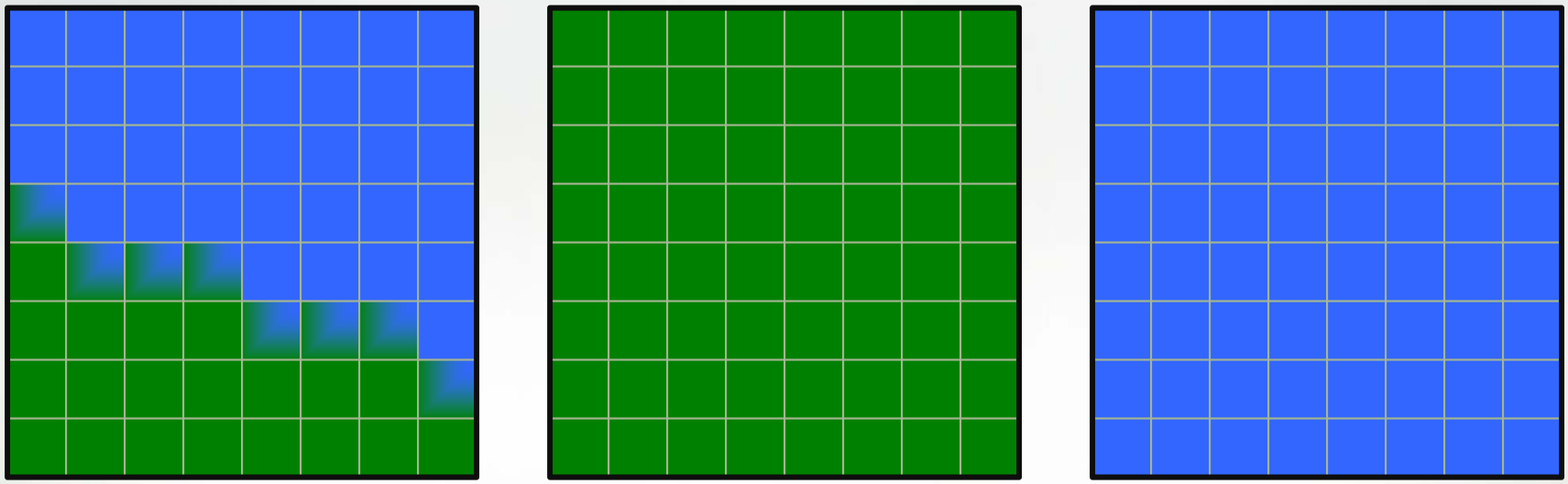
Materials

- LULESH models the behavior of materials within the elements



not all elements will contain all materials,
and some will contain combinations

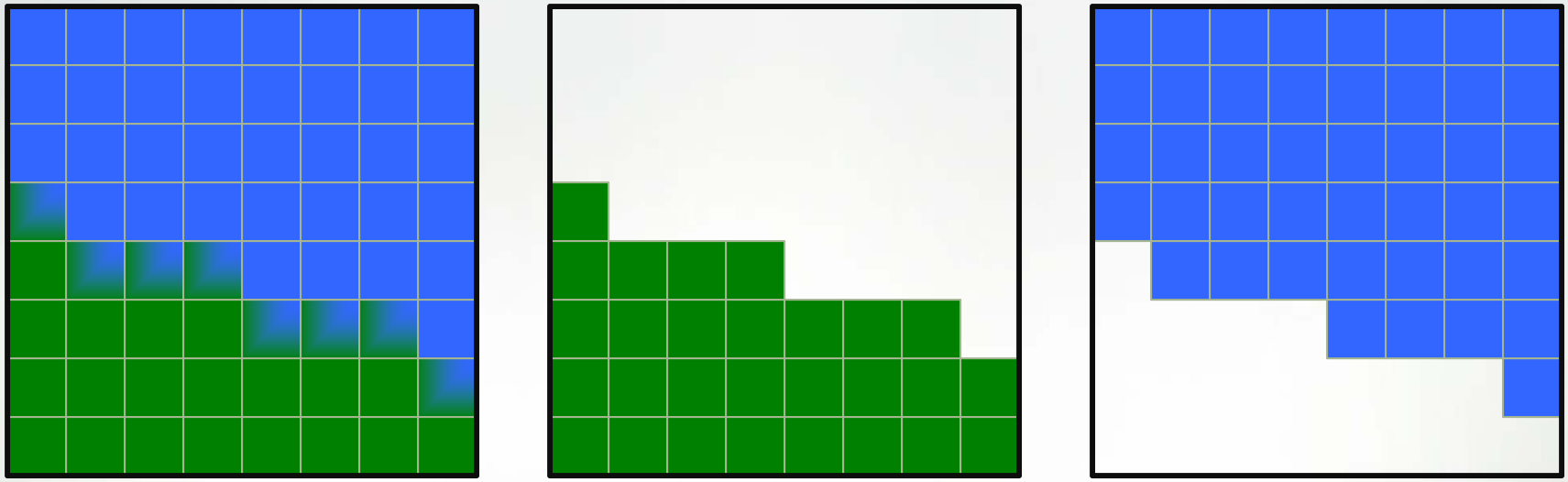
Materials



naïve approach: store all materials everywhere
(reasonable for LULESH, but not in practice)

```
const Mat1Elems = Elems,
      Mat2Elems = Elems;
```

Materials



improved approach: use sparse subdomains to only store materials where necessary

```
var Mat1Elems: sparse subdomain(Elems) = enumerateMat1Locs(),
    Mat2Elems: sparse subdomain(Elems) = enumerateMat2Locs();
```

The Physics

```
proc CalcKinematicsForElems(dxx, dyy, dzz, const dt: real) {
  // loop over all elements
  forall k in Elems {
    var b_x, b_y, b_z: 8*real,
        d: 6*real,
        detJ: real;

    //get nodal coordinates from global arrays and copy into local arrays
    var x_local, y_local, z_local: 8*real;
    localizeNeighborNodes(k, x, x_local, y, y_local, z, z_local);

    //get nodal velocities from global arrays and copy into local arrays
    var xd_local, yd_local, zd_local: 8*real;
    localizeNeighborNodes(k, xd, xd_local, yd, yd_local, zd, zd_local);
    var dt2 = 0.5 * dt; //wish this was local, too...

    local {
      //volume calculations
      const volume = CalcElemVolume(x_local, y_local, z_local);
      const relativeVolume = volume / volo.localAccess[k];
      vnew.localAccess[k] = relativeVolume;
      delv.localAccess[k] = relativeVolume - v.localAccess[k];

      //set characteristic length
      arealg.localAccess[k] = CalcElemCharacteristicLength(x_local, y_local,
                                                           z_local, volume);

      for param i in 1..8 {
        x_local[i] -= dt2 * xd_local[i];
        y_local[i] -= dt2 * yd_local[i];
        z_local[i] -= dt2 * zd_local[i];
      }

      CalcElemShapeFunctionDerivatives(x_local, y_local, z_local,
                                       b_x, b_y, b_z, detJ);

      CalcElemVelocityGradient(xd_local, yd_local, zd_local, b_x, b_y, b_z,
                               detJ, d);

    }

    // put velocity gradient quantities into their global arrays.
    dxx.localAccess[k] = d[1];
    dyy.localAccess[k] = d[2];
    dzz.localAccess[k] = d[3];
  }
}
```

The Physics

```
proc CalcKinematicsForElems(dxx, dyy, dzz, const dt: real) {
  // loop over all elements
  forall k in Elems {
    var b_x, b_y, b_z: 8*real,
        d: 6*real,
        detJ: real;

    //get nodal coordinates from global arrays and copy into local arrays
    var x_local, y_local, z_local: 8*real;
    localizeNeighborNodes(k, x, x_local, y,

    //get nodal velocities from global arra
    var xd_local, yd_local, zd_local: 8*rea
    localizeNeighborNodes(k, xd, xd_local, yd, yd_local, zd, zd_local);
    var dt2 = 0.5 * dt; //wish this was local, too...

    local {
      //volume calculations
      const volume = CalcElemVolume(x_local, y_local, z_local);
      const relativeVolume = volume / volo.localAccess[k];
      vnew.localAccess[k] = relativeVolume;
      delv.localAccess[k] = relativeVolume - v.localAccess[k];

      //set characteristic length
      arealg.localAccess[k] = CalcElemCharacteristicLength(x_local, y_local,
                                                           z_local, volume);

      for param i in 1..8 {
        x_local[i] -= dt2 * xd_local[i];
        y_local[i] -= dt2 * yd_local[i];
        z_local[i] -= dt2 * zd_local[i];
      }

      hapeFunctionDerivatives(x_local, y_local, z_local,
                              b_x, b_y, b_z, detJ);

      CalcElemVelocityGradient(xd_local, yd_local, zd_local, b_x, b_y, b_z,
                               detJ, d);
    }

    // put velocity gradient quantities into their global arrays.
    dxx.localAccess[k] = d[1];
    dyy.localAccess[k] = d[2];
    dzz.localAccess[k] = d[3];
  }
}
```

Representation-Independent Physics!

LULESH in Chapel

- physics code (all but ~25 lines) unchanged when switching…
  - …from 3D regular- vs. 1D irregular-mesh
  - …from dense vs. sparse materials elements representation
- great demonstration of domain maps, rank independent syntax
- LLNL application scientists notably impressed

Why We <3 LULESH

- Access to expert knowledge for a code that people actually care about
  - Tips on performance tuning
- Feedback on the language
- Ideas for new sparse data structures
- New challenges to the language

CRAY
THE SUPERCOMPUTER COMPANY

Episode VII

?

SC12
Salt Lake City, Utah

The Next Steps: PERFORMANCE

- LULESH-specific
  - tuples
  - array-of-structs vs. struct-of-arrays
- General
  - Reductions
    - ~50% of the per cycle time at 64 locales
  - Communication optimizations
    - aggregation
    - overlap
    - floating point atomics (when AMOs not available)

Will there be and Episode VII?

- LULESH-specific **[NOT FUNDED]**
  - tuples
  - array-of-structs vs. struct-of-arrays

- General **[POTENTIALLY FUNDED]**
  - Reductions
    - ~50% of the per cycle time at 64 locales
  - Communication optimizations
    - aggregation
    - overlap
    - floating point atomics (when AMOs not available)

Benchmark Sources

LULESH:

- in Chapel release: $CHPL_HOME/examples/benchmarks/lulesh/
- In Subversion tree:
  - Elegant version:
    - https://chapel.svn.sourceforge.net/svnroot/chapel/trunk/test/release/examples/benchmarks/lulesh/
  - Performance studies version:
    - https://chapel.svn.sourceforge.net/svnroot/chapel/trunk/test/studies/lulesh/bradc/lulesh-dense.chpl

(Recipes for compiler/execution/environment options for our performance results available by request)

Chapel at SC12 (see chapel.cray.com/events.html for details)

- ✓ **Sun:** Chapel tutorial (8:30am)
- ✓ **Mon:** 3rd Annual Chapel Users Group (CHUG) Meeting
- ✓ **Tues:** HPC Challenge BoF (12:15pm)
- ✓ **Wed:** Chapel Lightning Talks BoF (12:15pm)
- ✓ **Wed:** Chapel talk at KISTI booth (~3-4pm)
- ✓ **Wed:** HPCS BoF (5:30pm)
- ➢ **Wed:** Proxy Applications for Exascale BoF (5:30pm)
- • **Thurs:** HPC Educators Forum on Chapel (1:30pm)

Resources For After Today

Chapel project page: http://chapel.cray.com

- overview, papers, presentations, language spec, …

Chapel SourceForge page: https://sourceforge.net/projects/chapel/

- release downloads, public mailing lists, code repository, …

IEEE TCSC Blog Series:

- *Myths About Scalable Parallel Programming Languages*

Mailing Lists:

- chapel_info@cray.com: contact the team
- chapel-users@lists.sourceforge.net: user-oriented discussion list
- chapel-developers@lists.sourceforge.net: dev.-oriented discussion
- chapel-education@lists.sourceforge.net: educator-oriented discussion
- chapel-bugs@lists.sourceforge.net/chapel_bugs@cray.com : public/private bug forum