EAVL

Dave Pugmire, Jeremy Meredith,
Sean Ahern, Rob Sisneros

ORNL, NCSA

CScADS
July/August 2012

Overview

- Funded under FY2011, FY2012 exascale-focused ORNL LDRD call for proposals
  - Jeremy Meredith (PI), Sean Ahern, David Pugmire, and Robert Sisneros (postdoc, now at NCSA)
- Exploring exascale vis/analysis issues
  - concurrency, scalability, computational and memory efficiency, data model, heterogeneous architectures
- "Extreme-scale Analysis and Visualization Library"
  - Note: originally "Exascale"

What are the exascale problems?

- Concurrency will rise 40,000x to 400,000x
- Memory will rise by only 100x
- I/O subsystem will be slower and smaller
- Simulation codes are evolving with new mesh and data models
- In situ analysis and visualization

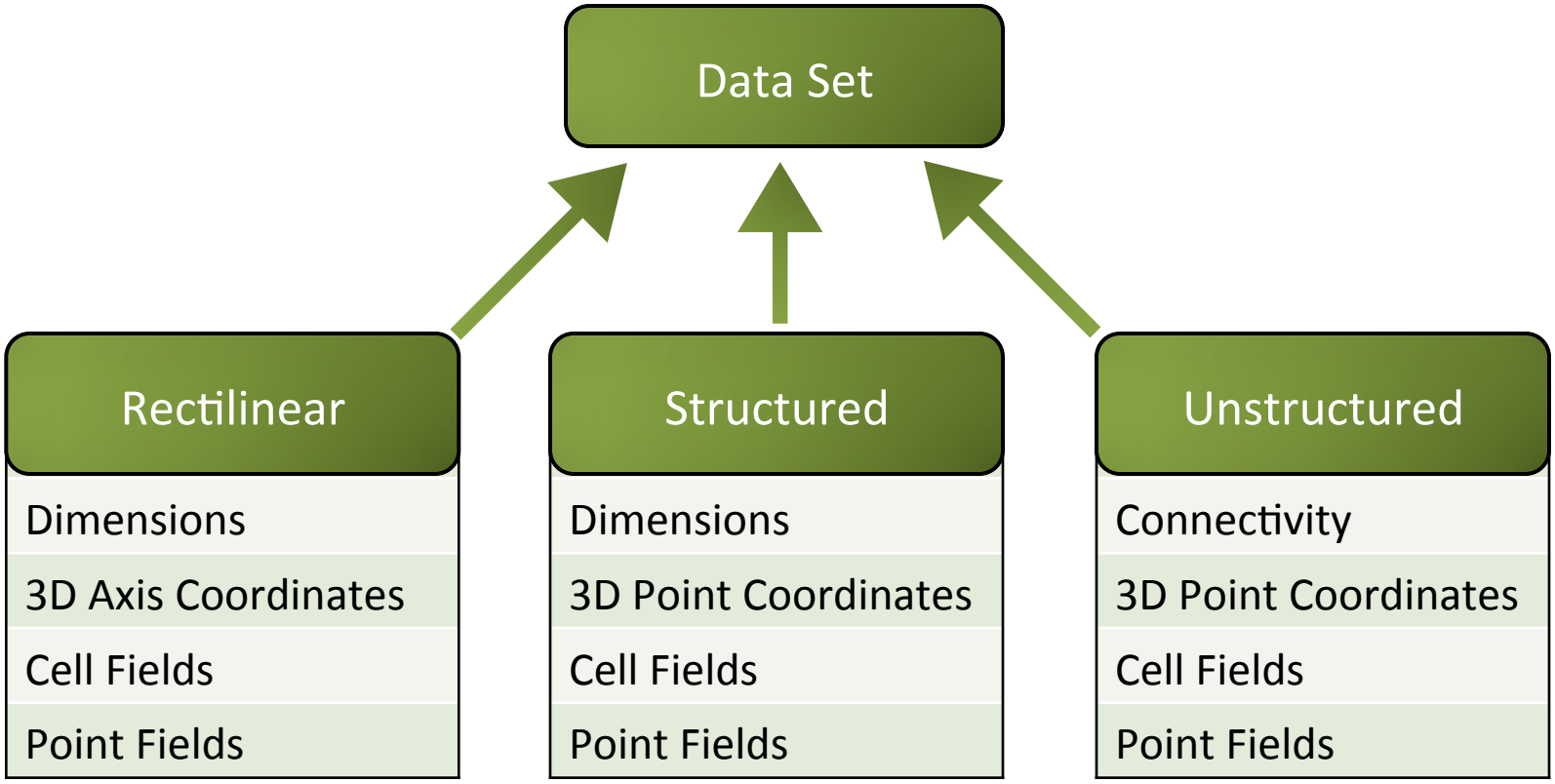
Yes, these are a broad set of issues…..

How are we trying to solve them?

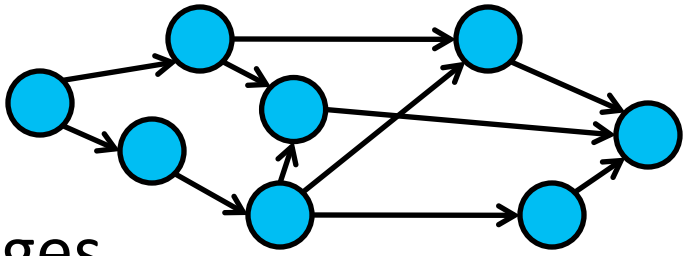
- New data model
  - Support for wider range of data
- Increased efficiency
  - Computational and memory efficiency
- New avenues for scalability for next generation architectures
  - MPI, GPU, OpenMP, MIC

Data Modeling Challenges

A Traditional Data Set Model

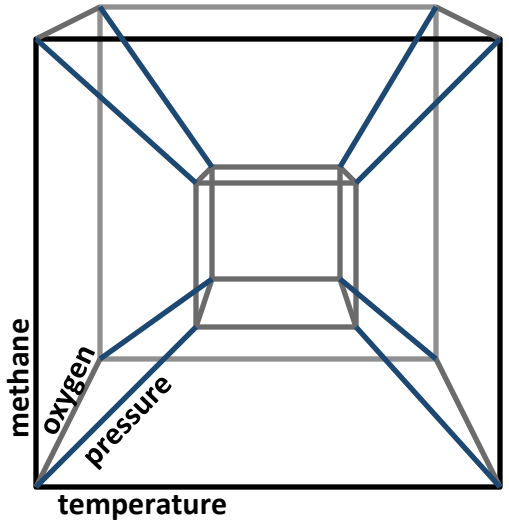


Challenge: Non-Physical Data Analysis

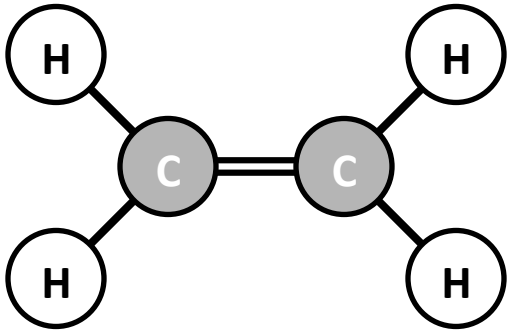


- Graph Data
  - topologically 0D vertices, 1D edges
  - non-spatial; storing X/Y/Z values is wasted space

- Pure Parameter Studies
  - e.g. reaction rate of combustion
    - FOUR "spatial" dimensions
      - e.g. methane concentration vs oxygen concentration vs temperature vs pressure
    - more complex reaction → higher dimensionality



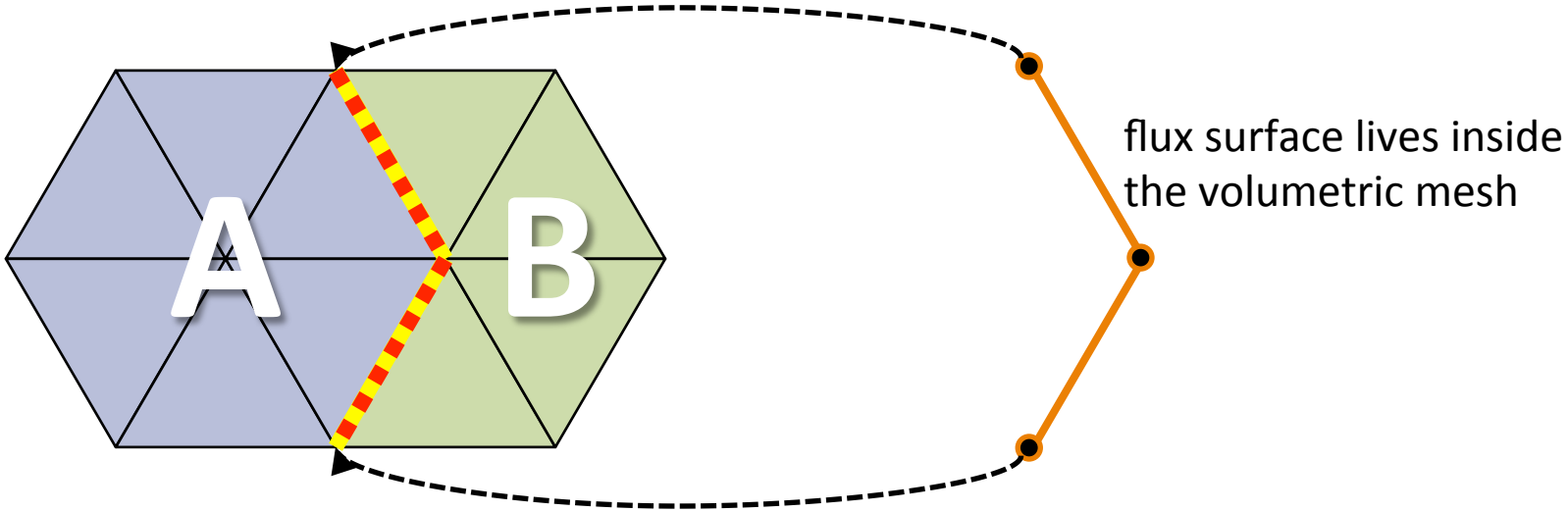
Challenge: Molecular Data
(e.g., LAMMPS, VASP)



BondStr	AtomicNum
1	6
1	6
1	1
1	1
2	1
	1

- To represent using vtkPolyData or vtkUnstructuredGrid:
  - VTK_VERTEX cells for the atoms
  - VTK_LINE cells for the bonds
- Any field data must exist on both element types
  - Not only inefficient:
    - dummy bond strengths on the atoms?
    - dummy atomic numbers on the bonds?
  - But also incorrect:
    - e.g. average(BondStrength) uses dummy values from atoms?

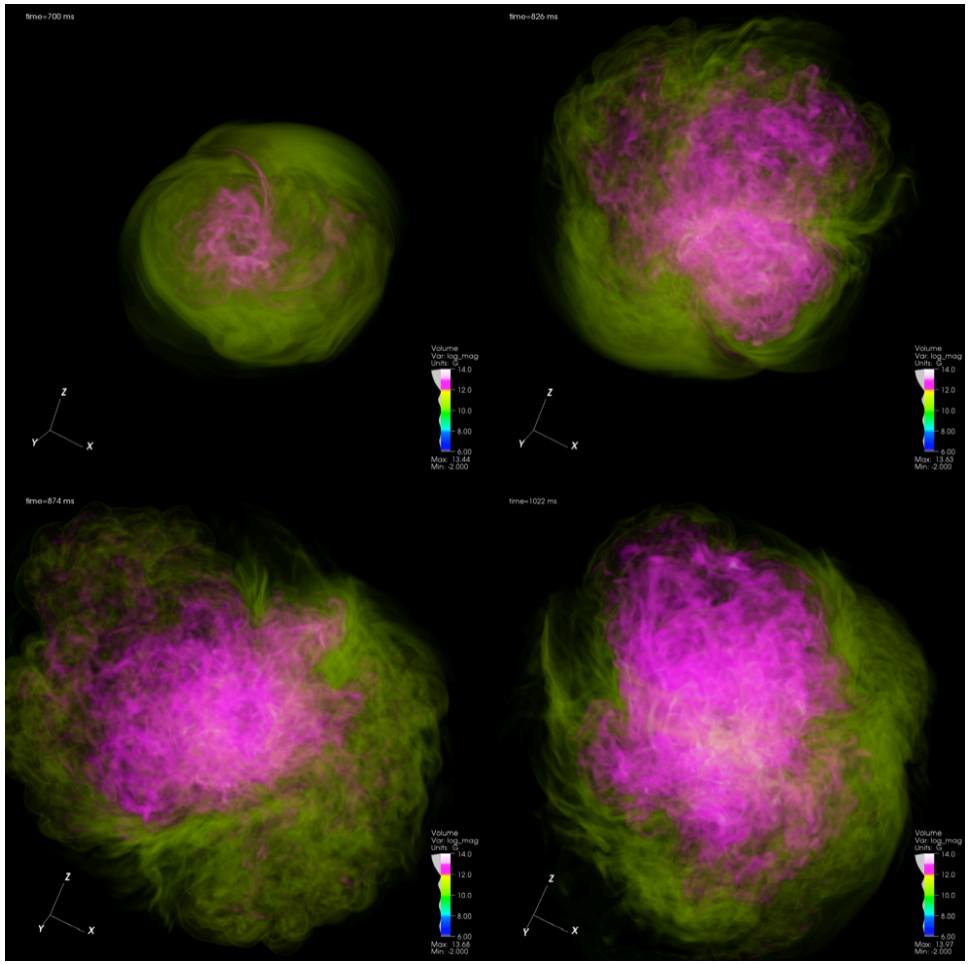
Challenge: Side Sets
(e.g. Exodus, flux surfaces)



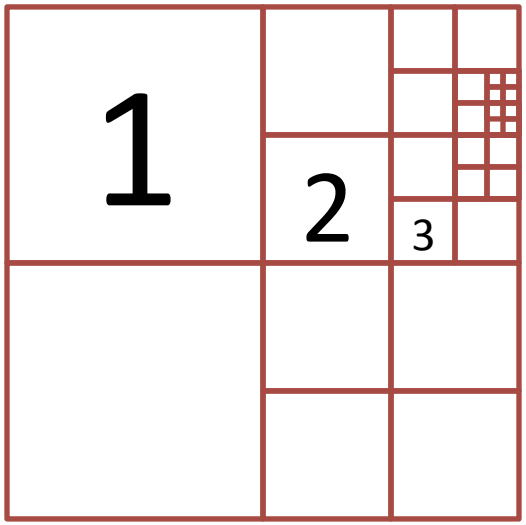
- The flow from A to B is defined on a set of faces
- The flux variable is defined *only* on those faces
  - do you combine them into a single mesh?
    - waste space on dummy values, potentially introducing errors
  - or create a separate mesh and lose the mapping info?
    - horribly expensive and error-prone to recalculate mapping

Challenge: High Dimensionality
(e.g. GenASiS)

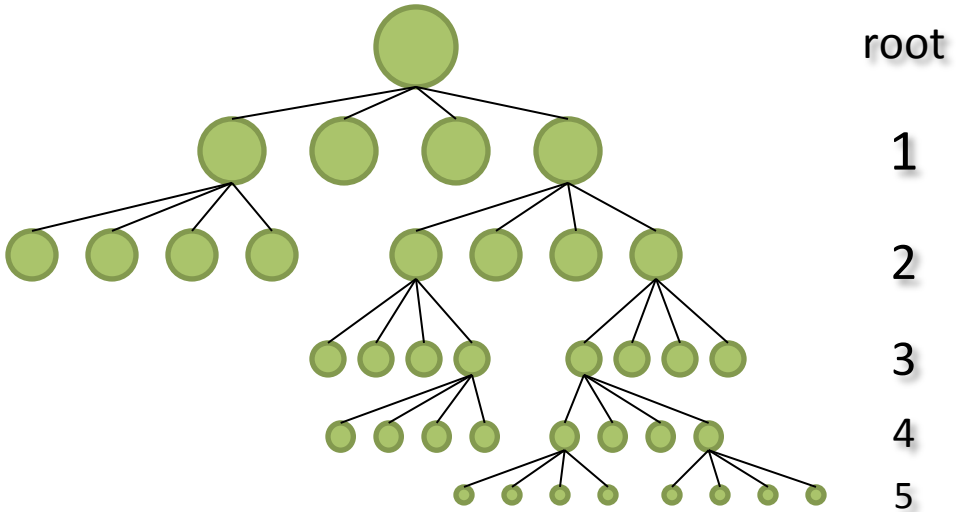
- Seven (or eight) dimensional mesh
  - $f(x,y,z,\Theta,\phi,\lambda,F)=E$, plus time



Challenge: Unique Mesh Topologies
(e.g. MADNESS)



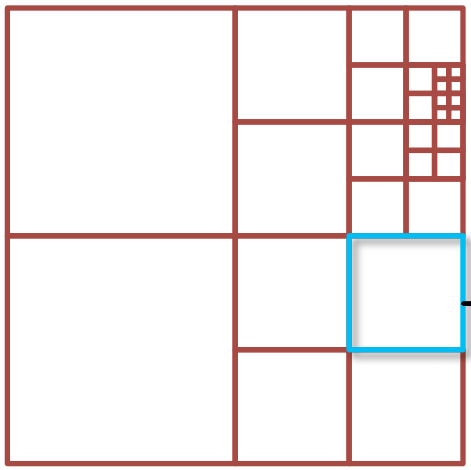
spatial structure



internal tree representation

- MADNESS does not have a traditional mesh
  - Just a quad-tree with polynomial coefficients
  - Up to 30 refinement levels / tree depth

Challenge: Very High Order Fields
(e.g. MADNESS)



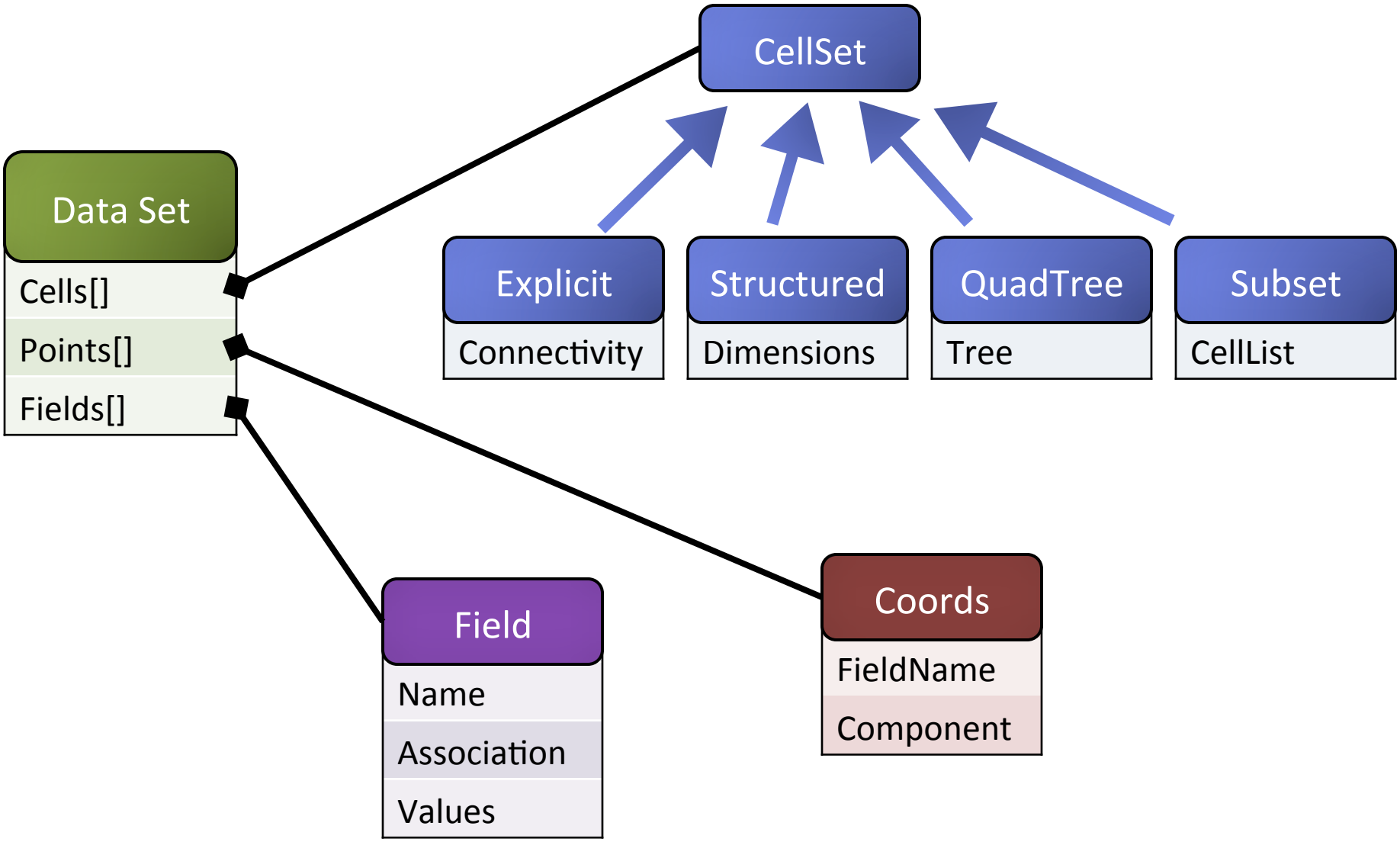
(example with K=3, dim=2)

0.834	0.592	0.003
0.592	0.003	0.010
0.003	0.010	0.007

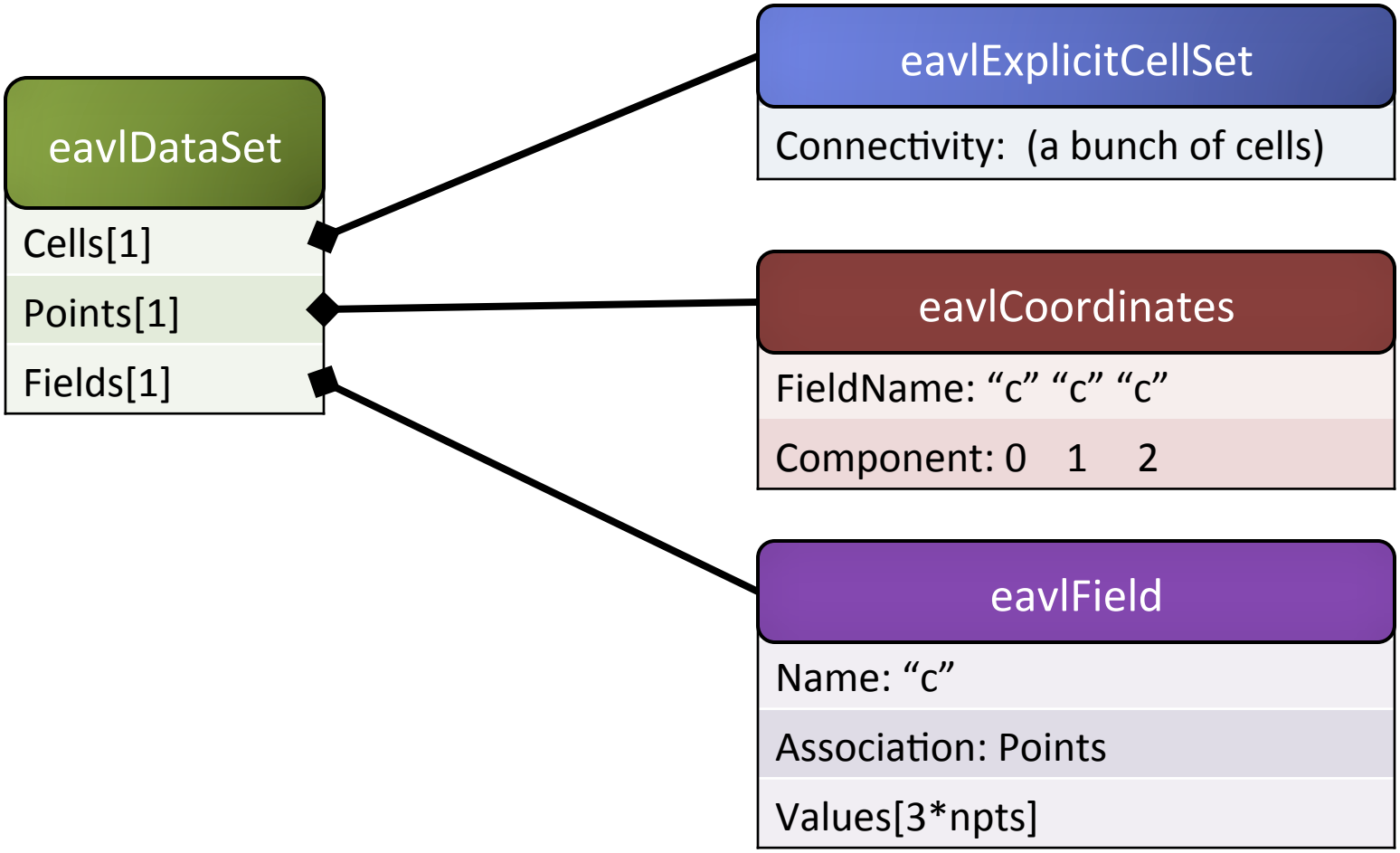
- Legendre polynomial series at each tree node
  - Each tree node has K^{dim} coefficients
  - K can be up to approx. 20
    - i.e. 400 coeffs per tree node in 2D, 8000 in 3D

The EAVL Data Model

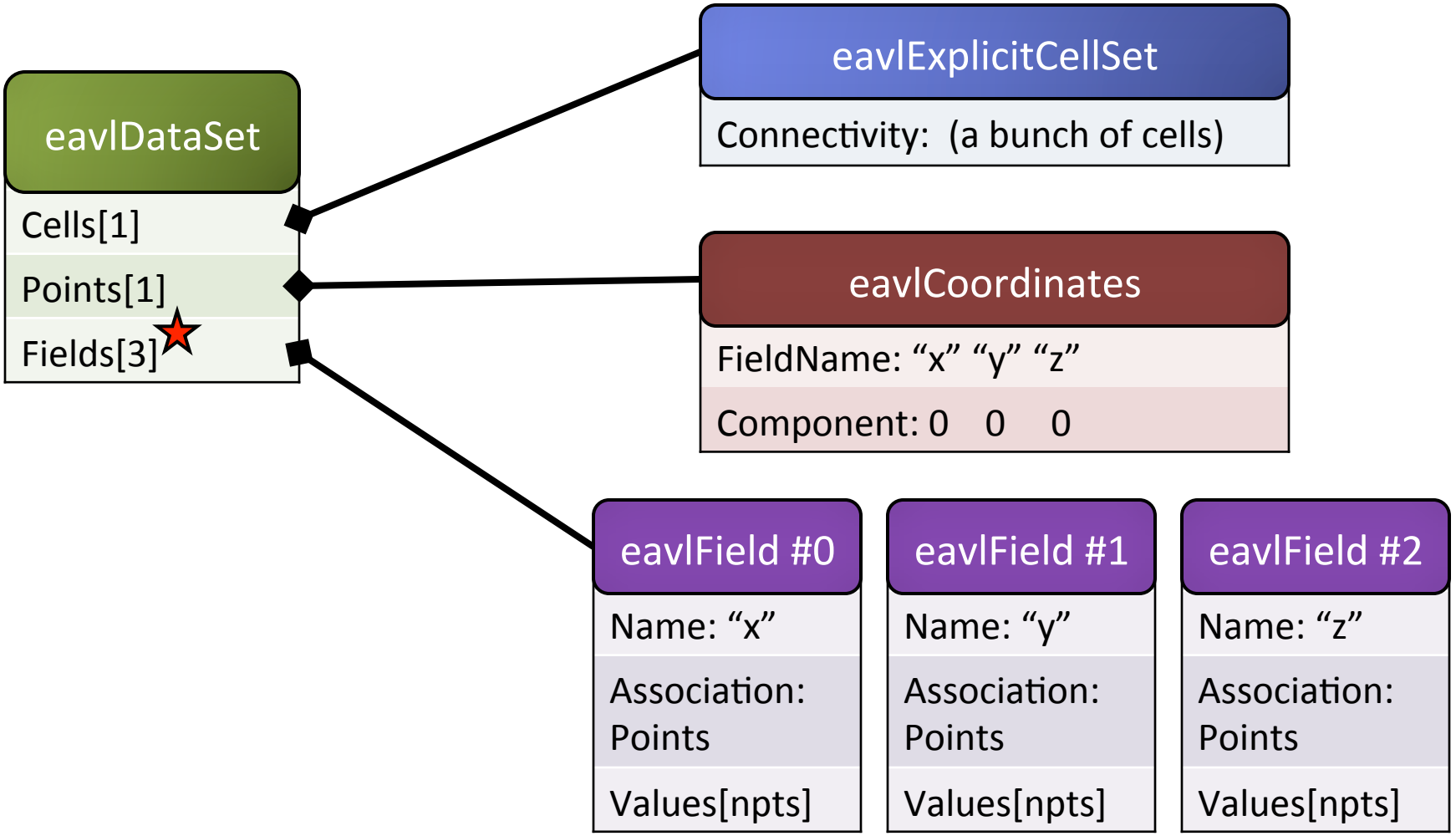
The EAVL Data Set Model



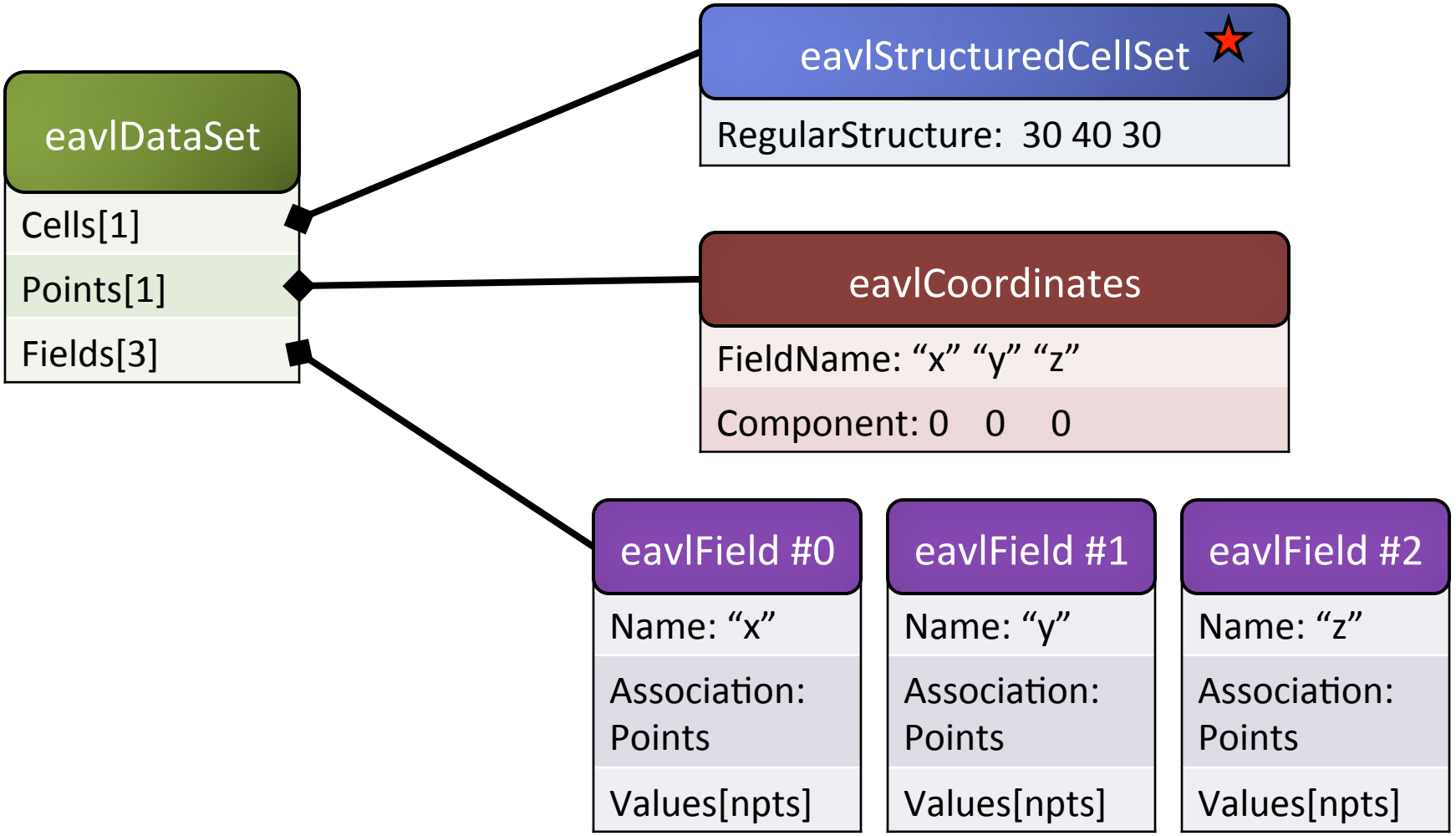
Example: An Unstructured Grid
(with interleaved coordinates)



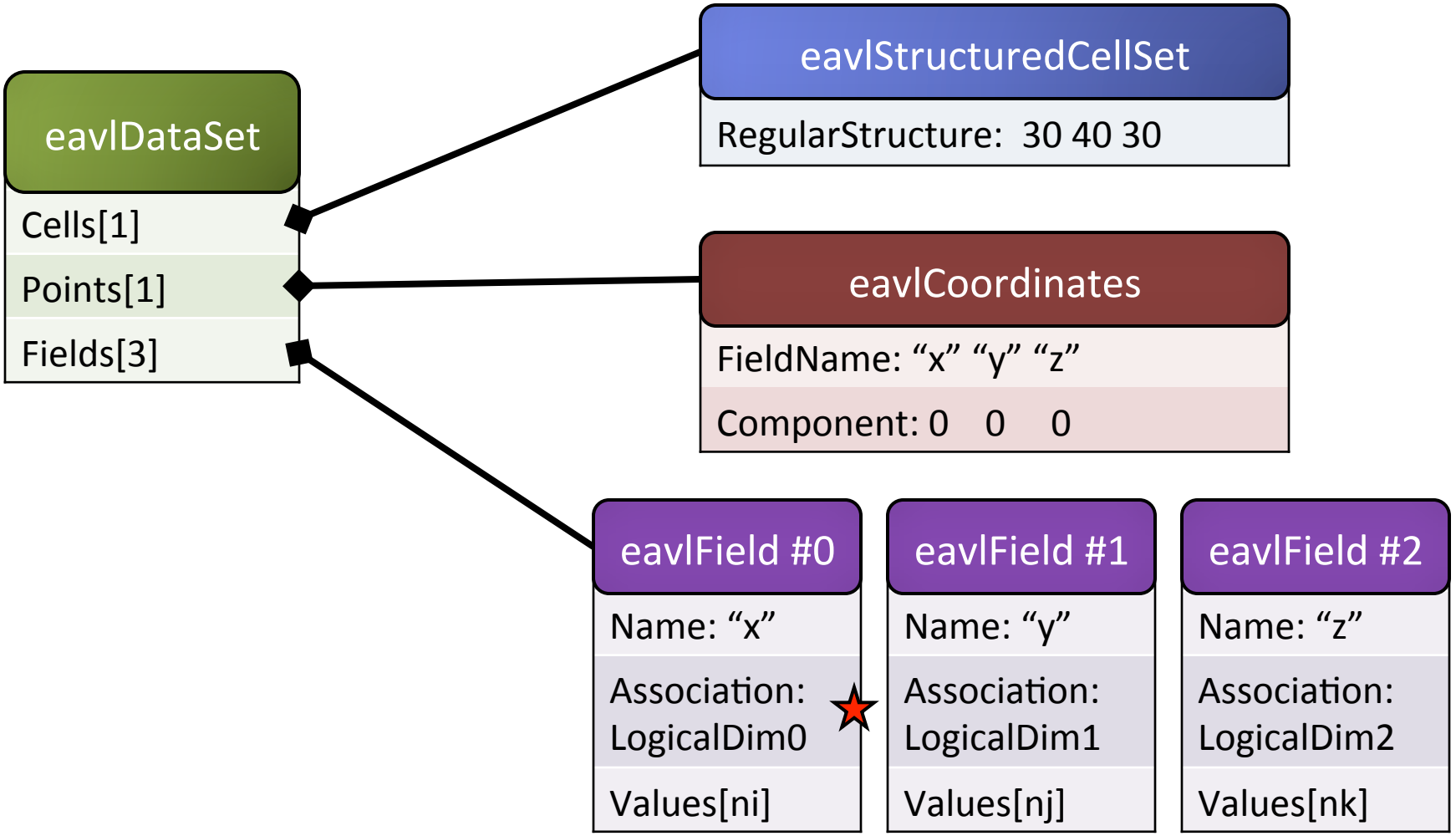
Example: An Unstructured Grid
(with separated coordinates)



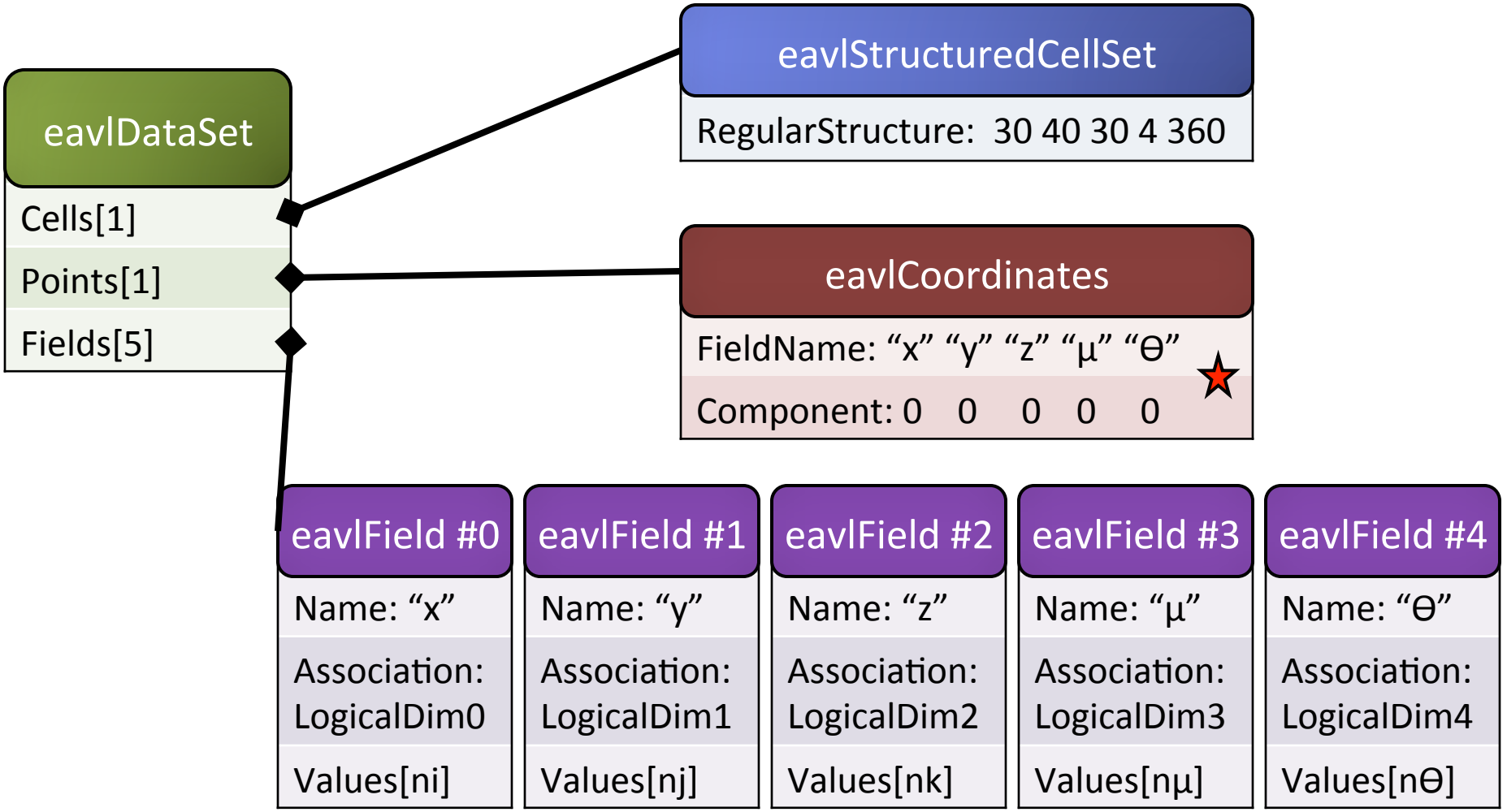
Example: A Curvilinear Grid



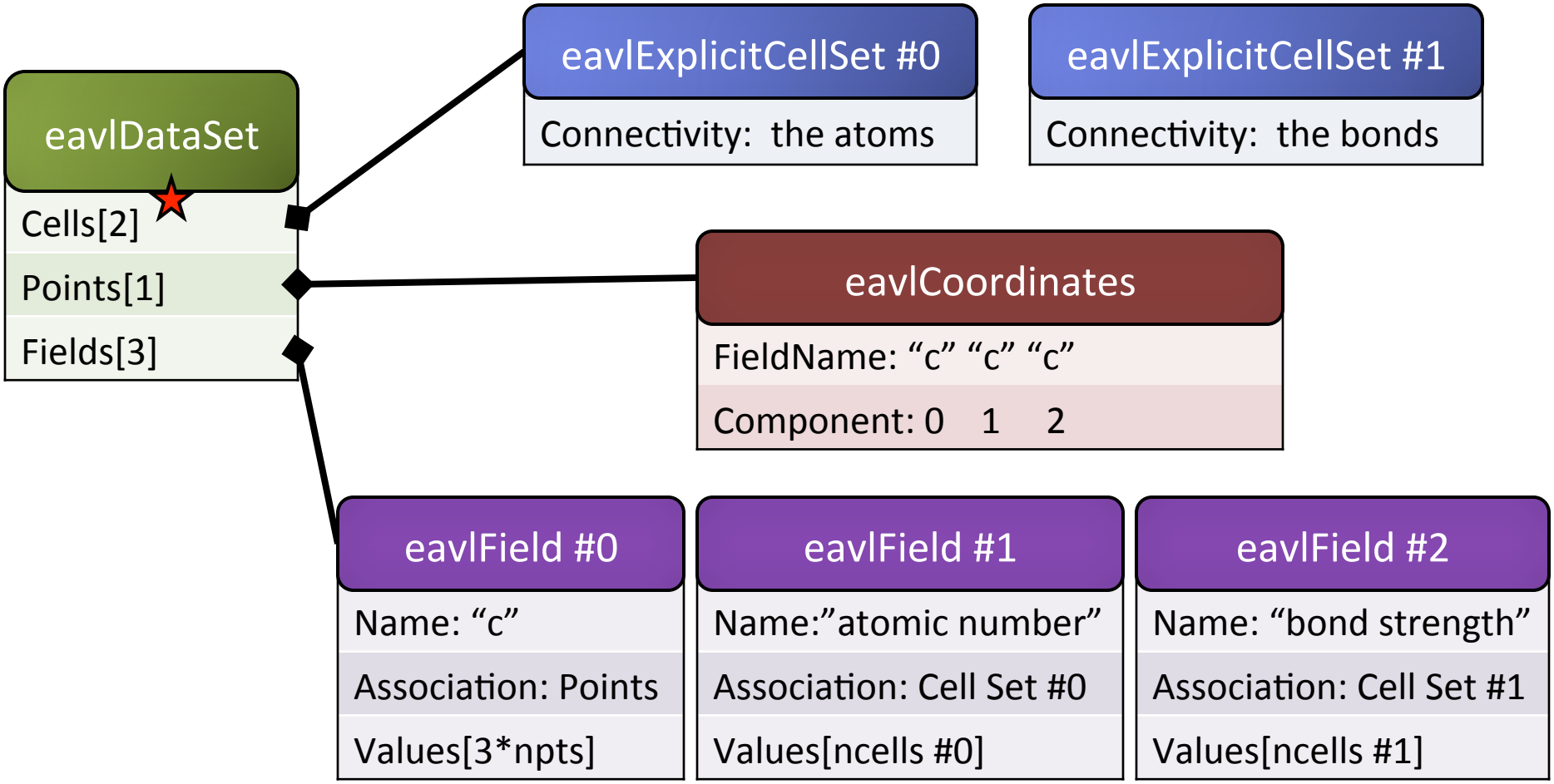
Example: A Rectilinear Grid



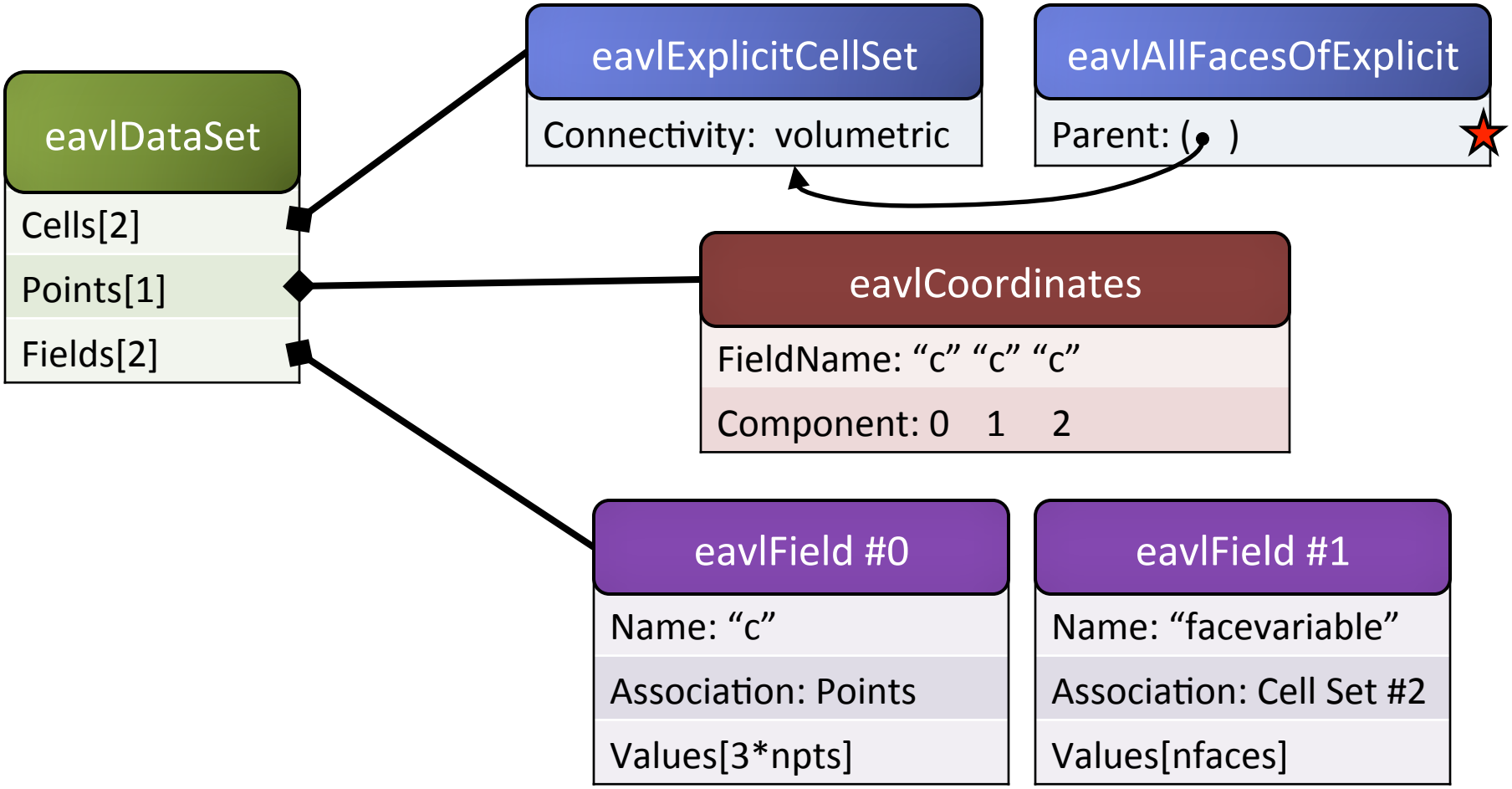
Example: High-Dimensional Grid



Example: Molecular Data



Example: Face-centered Data



Filtering in EAVL

Data flow networks in EAVL (or not)

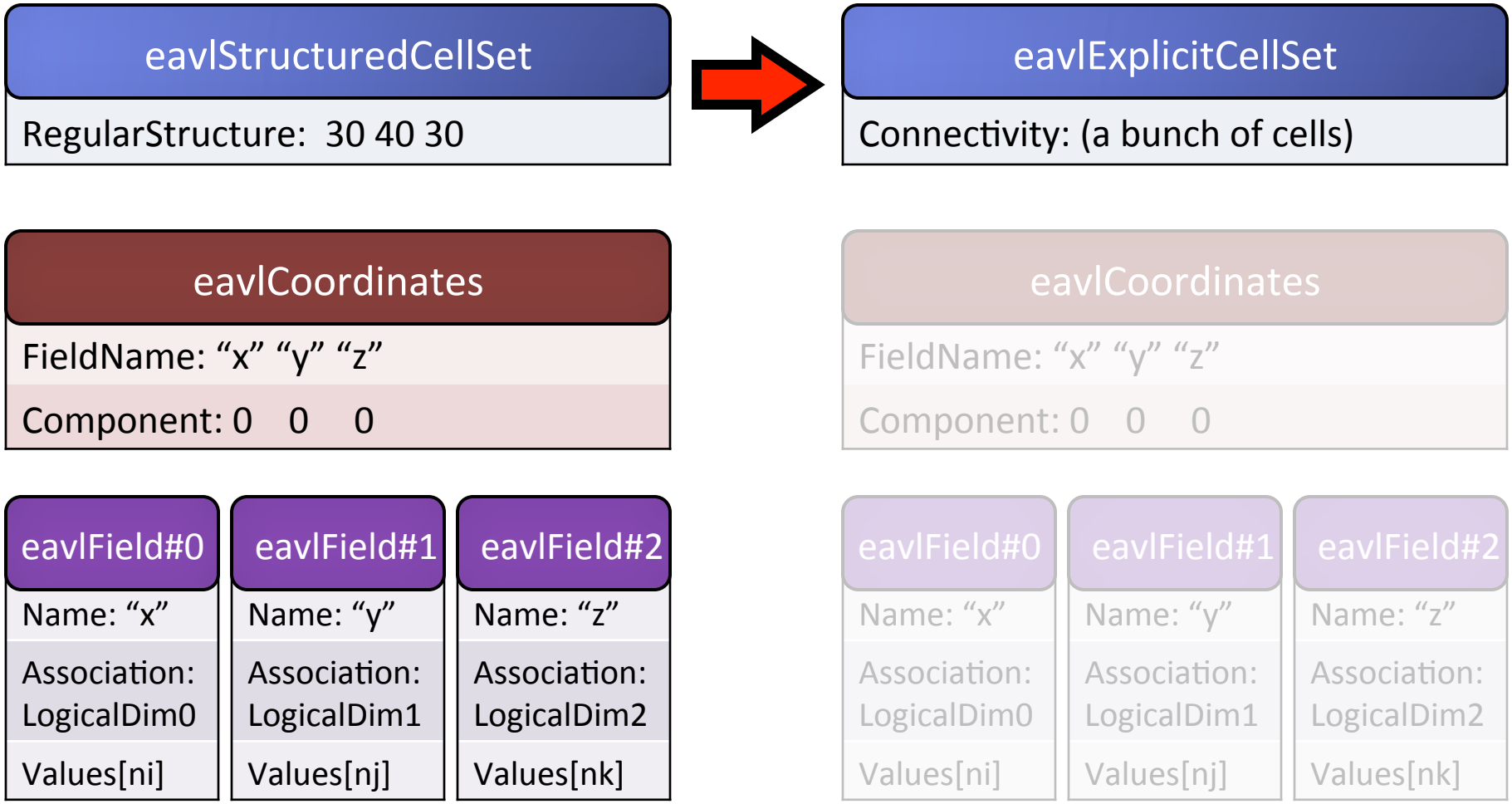
- A “Filter” is a stage in a data flow network
  - Creates a new data set from an old one
- Many operations do not change a mesh structure (assuming data model is sufficiently descriptive)
  - Arithmetic expressions: only modifies fields
  - External facelist: points and structure remain
  - Feature edges: just a new cell set with old points
  - Smooth, displace, elevate: only modify coordinates
- So: **eavlMutator** is an alternative to eavlFilter
  - Modifies a data set in-place

eavlMutator

- In-place data set modification
- Support for **destructive** in-place operation
  - free memory as you go
- Execute multiple mutators simultaneously on the same data set (barring conflicts)
  - e.g. displace (coords) + threshold (cells) concurrently
- How about data flow network support?
  - encapsulate an eavlMutator through a eavlFilterFromMutator facade
- Of course, some operations are natively eavlFilters

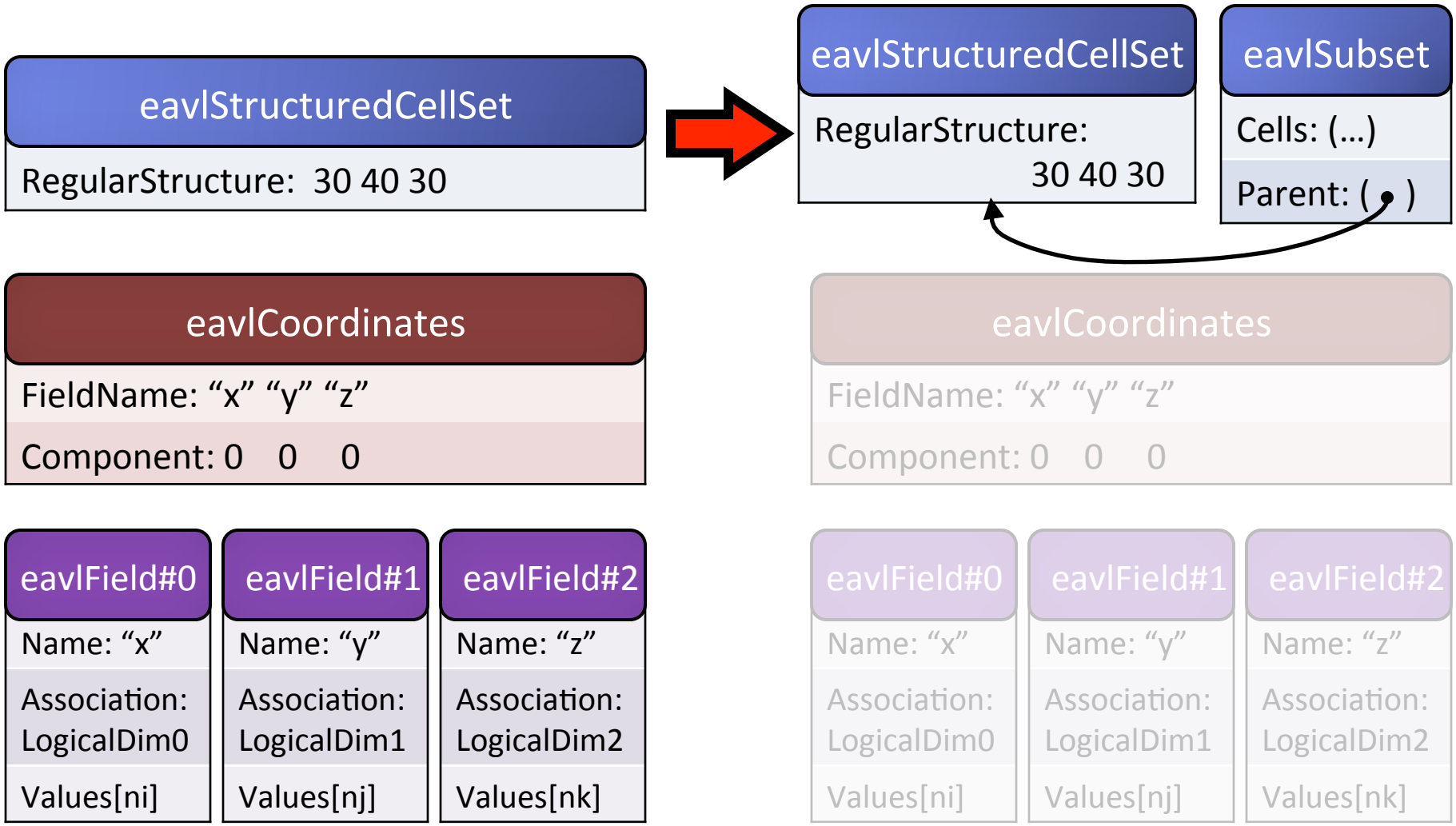
Example: Thresholding an RGrid (a)

- Explicit cells can be combined with structured coordinates.



Example: Thresholding an RGrid (b)

- A second Cell Set can be added which refers to the first one



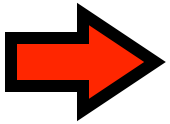
Example: Elevating a Structured Grid

- No problem-sized data modifications.
  - Interleaved and separated coordinates can be used simultaneously.

eavlStructuredCellSet
RegularStructure: 30 40

eavlStructuredCellSet
RegularStructure: 30 40

eavlCoordinates
FieldName: “c” “c”
Component: 0 1



eavlCoordinates
FieldName: “c” “c” “val”
Component: 0 1 0

eavlField#0
Name: “c”
Association: Points
Value[2*npts]

eavlField#1
Name: “val”
Association: Points
Values[npts]

eavlField#0
Name: “c”
Association: Points
Value[2*npts]

eavlField#1
Name: “val”
Association: Points
Values[npts]

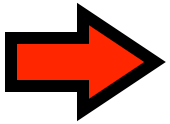
Example: Elevating a Regular Grid

- No problem-sized data modifications.
  - Some axes on logical dims, with others on the points.

eavlStructuredCellSet
RegularStructure: 30 40

eavlStructuredCellSet
RegularStructure: 30 40

eavlCoordinates
FieldName: “x” “y”
Component: 0 0



eavlCoordinates
FieldName: “x” “y” “val”
Component: 0 0 0

eavlField#0	eavlField#1	eavlField#2
Name: “x”	Name: “y”	Name: “val”
Association: LogicalDim0	Association: LogicalDim1	Association: Points
Values[ni]	Values[nj]	Values[npts]

eavlField#0	eavlField#1	eavlField#2
Name: “x”	Name: “y”	Name: “val”
Association: LogicalDim0	Association: LogicalDim1	Association: Points
Values[ni]	Values[nj]	Values[npts]

Dealing With Concurrency

Data Parallelism for Developers

- Functor + iterator paradigm
- Iteration patterns for mesh topologies
- CUDA + OpenMP execution back-ends

Functor + Iterator Approach

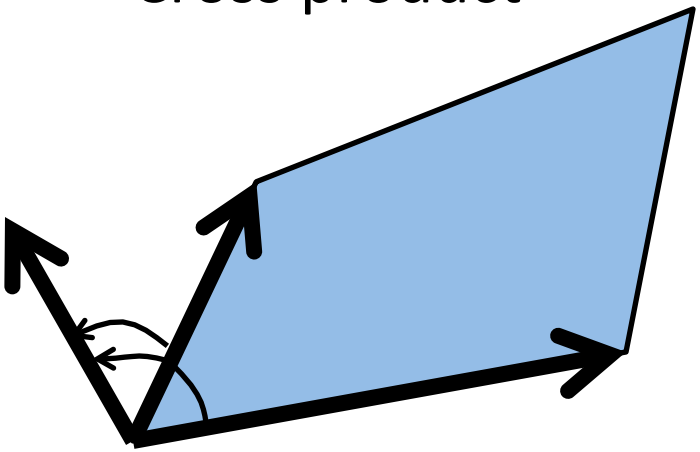
```
template <class T> void
CellToCellBinaryOp<T>(Field &a,
                      Field &b,
                      Field &c
                      T &f)
{
   for_each(i)
       f(a[i],b[i],c[i]);
}
```

```
struct Divide
{
   void operator()(float &a,
                   float &b,
                   float &c)
   {
      c = a / b;
   }
};
```

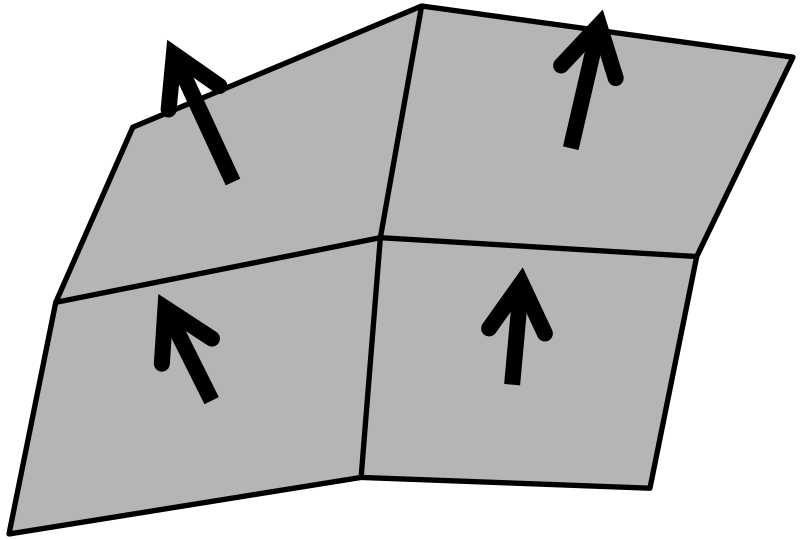
```
void CalculateDensity(...)
{
  //...
  CellToCellBinaryOp(mass, volume, density, Divide());
}
```

Example: Surface Normal

- For each 2D cell (i.e. each polygon):
  - Get three adjacent points
  - Pair-wise vector subtract
  - Cross product

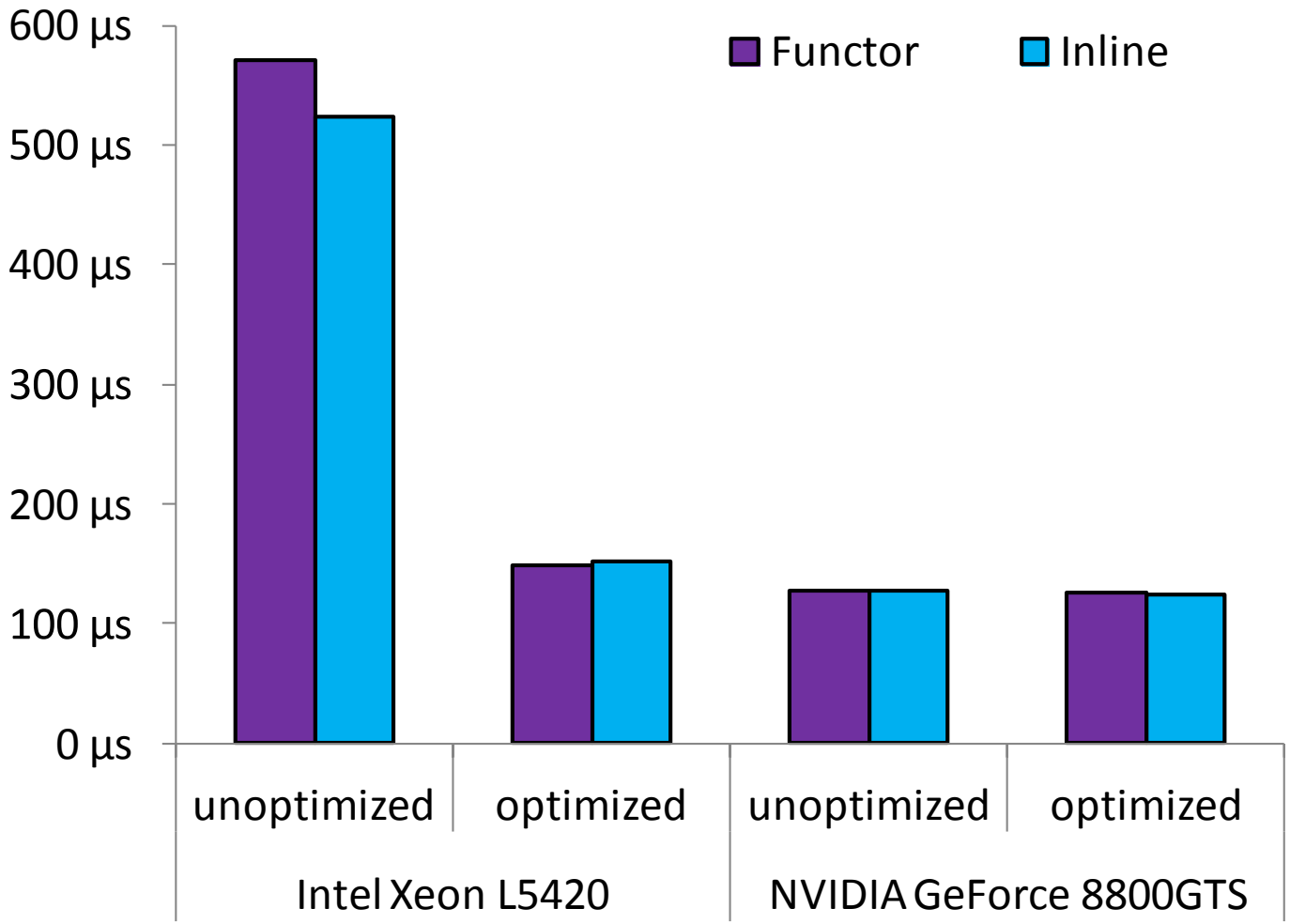


- Data-parallel:
  - Repeat for all cells



```
NodeToCellOp3::ExecuteCPU()
NodeToCellOp3::ExecuteGPU()
```

Functor Efficiency on CPU and GPU

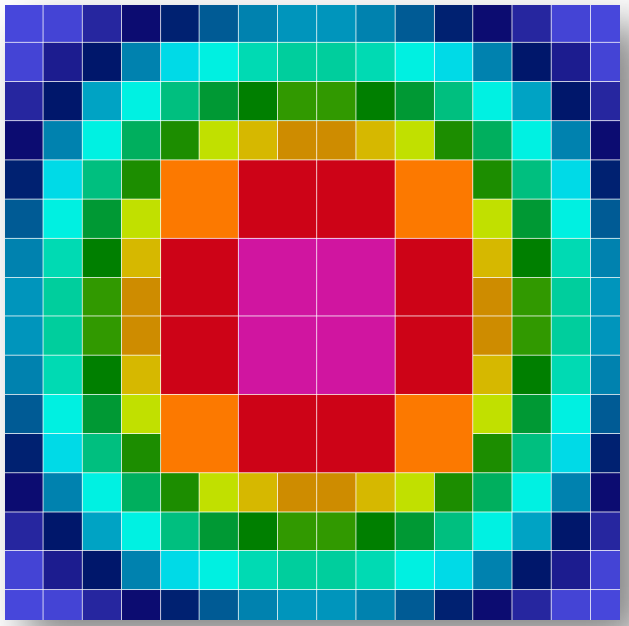


- Data: *noise.silo*
- Surface normal

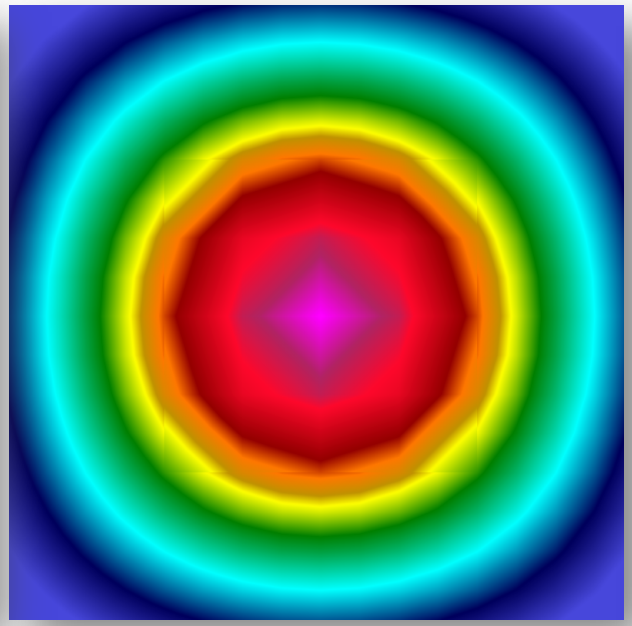
A Few Results

High-Order Quadtree

- MADNESS quadtree
- Can tesselate during pipeline
- Or let high-order fields make it to rendering
  - don't have to do explicit tesselation



1 value/cell (piecewise const)

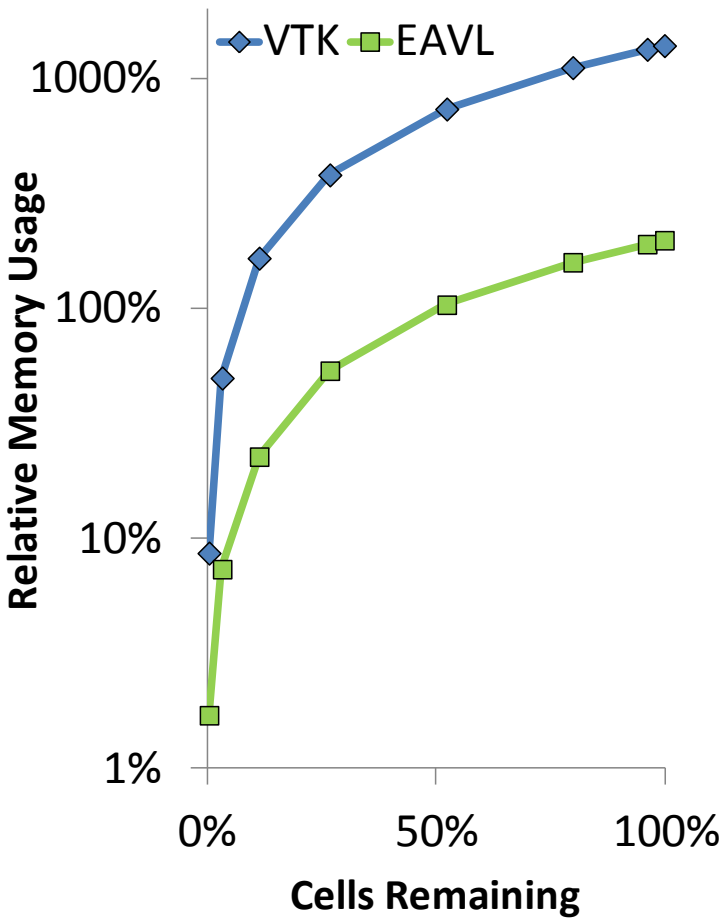


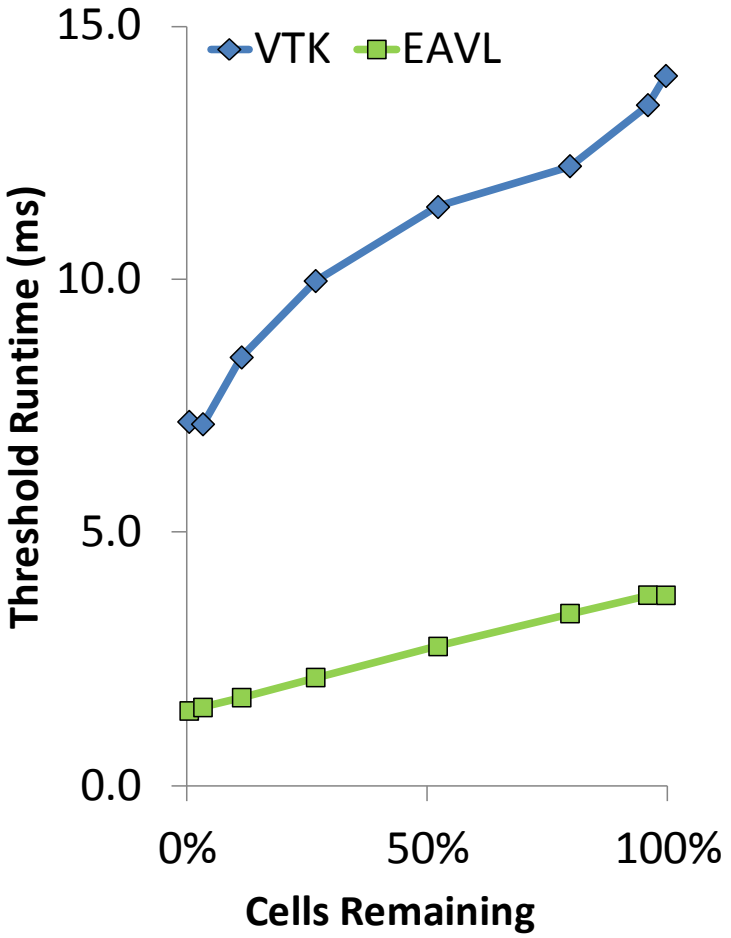
1 value/node (bilinear)

9 values/cell (biquadratic)

Regular Grid Threshold

- More memory efficient
- More computationally efficient





Elevating 2D → 3D

	rectilinear		structured		unstructured	
	before	after	before	after	before	after
EAVL	11 kB	11 kB	21 kB	21 kB	17 kB	17 kB
VTK	11 kB	21 kB	26 kB	21 kB	19 kB	17 kB

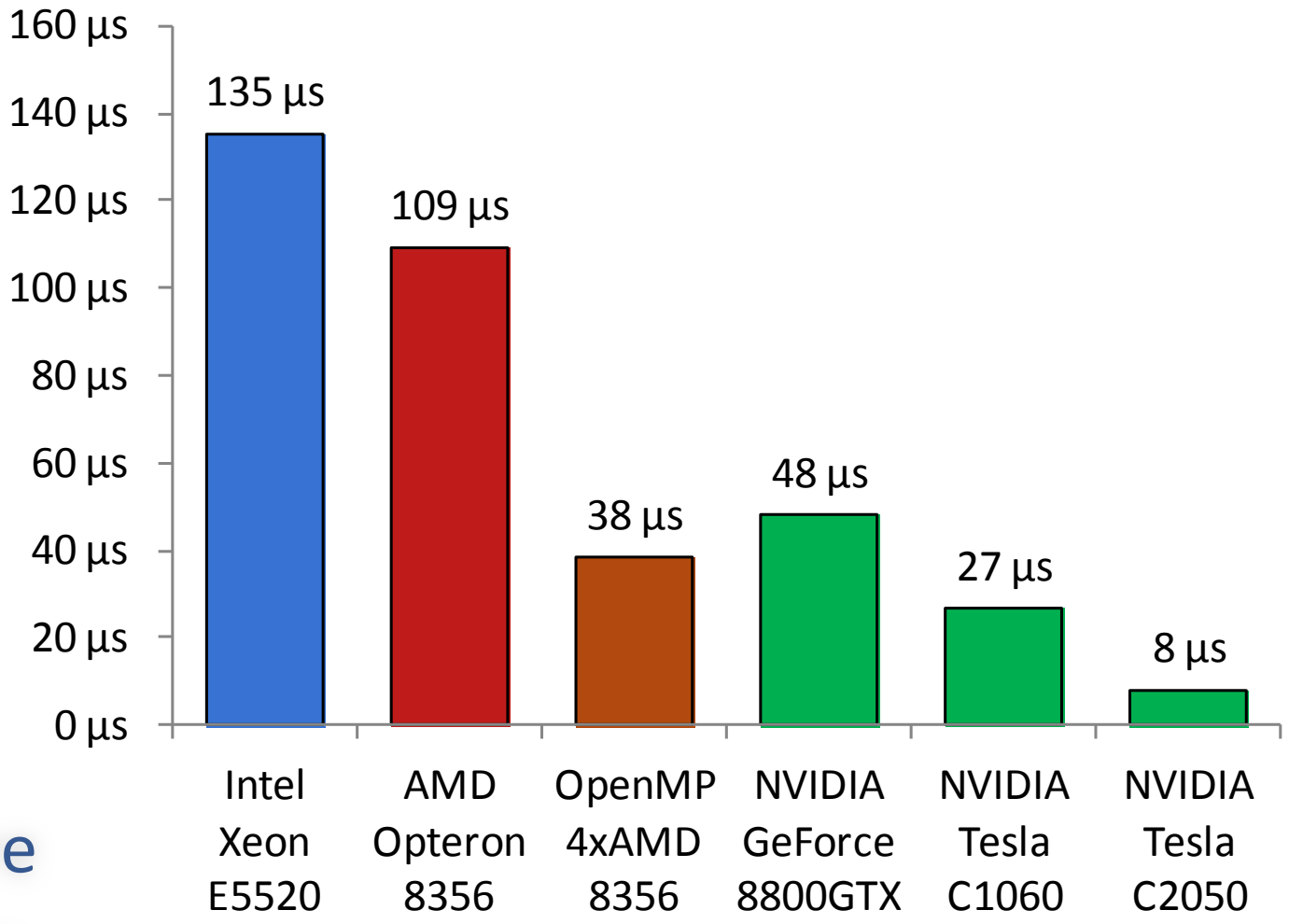
- EAVL:
  - any operation only changes mesh meta-data
  - constant-time operation
- VTK:
  - linear-time operation
  - rectilinear must become curvilinear
  - curvilinear and unstructured: extra field copied over wasted 3rd coordinate array (and then deleted)

Face Data for a 100^3 Structured Grid

- Options for a data model without face data:
  a) store as a 2.9M-polygon explicit data set
  b) store as 300 regular grids
     (but lose association as a single data set)
- In EAVL, can be both memory-efficient and correct

Representation	Memory for Grid + 1 Scalar Field
Single VTK polydata	74.8 MB
300 VTK rectilinear grids	12.3 MB
Single EAVL regular grid	11.8 MB

Data Parallelism: Surface Normal

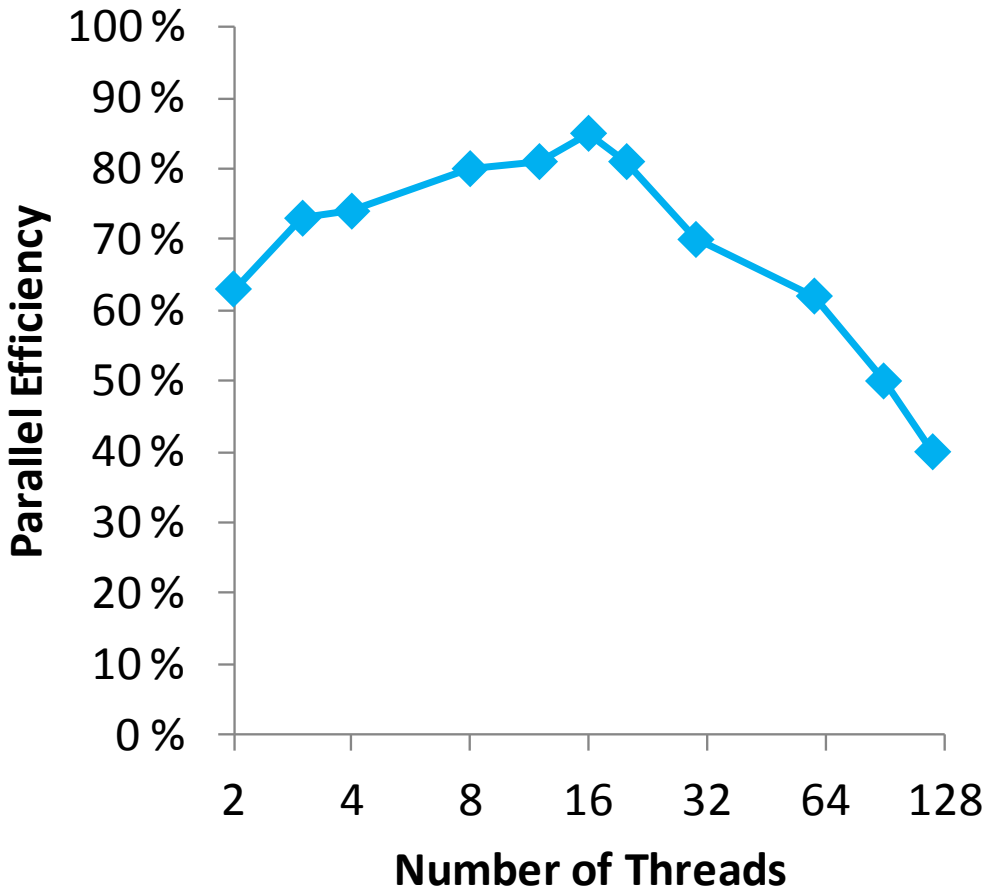


- Data: *noise.silo*
- Single-core
- Multi-core
- GPU

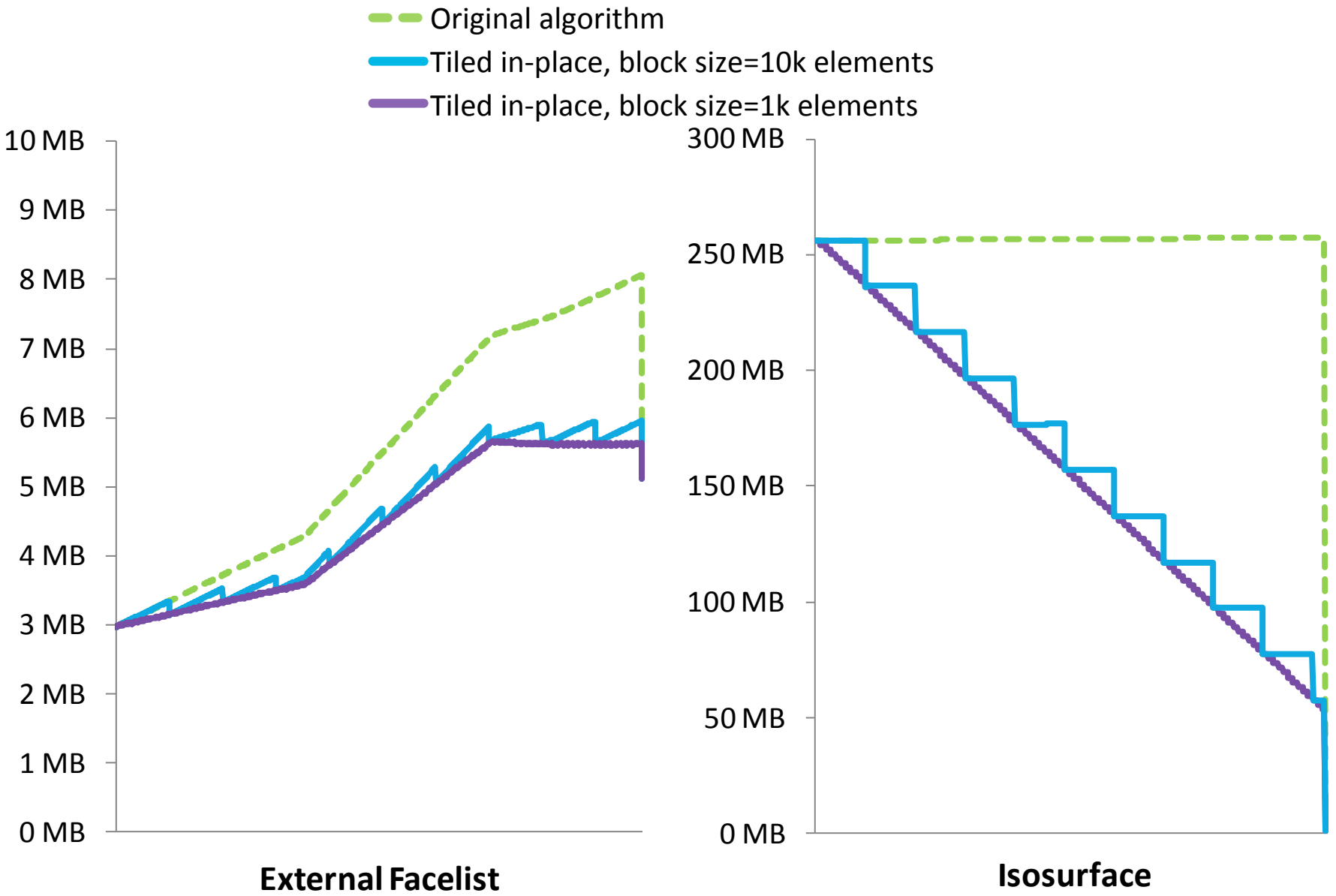
Data Parallelism: Surface Normal

- Target Intel® MIC Architecture SDP
- Used OpenMP, Intel compiler

- Data: *noise.silo*
- Surface normal

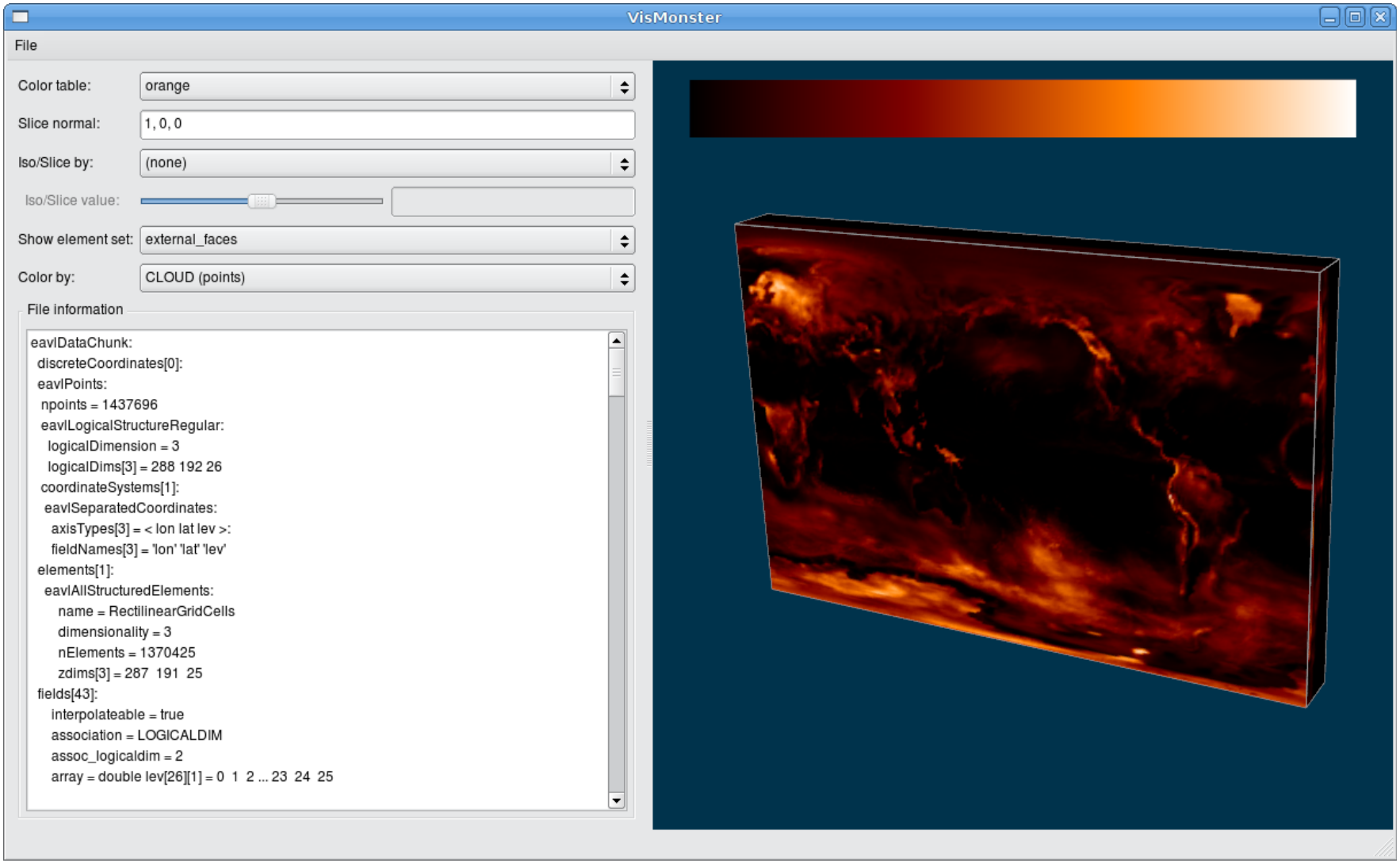


Experiment: Chunking and Destructive In-Place Operations



EAVL/VisMonster

- Fixed-pipeline demonstration GUI



Plans and Status

- Investigating deployment
  - prototype GUI
  - within VisIt
  - in situ with simulation code via ADIOS
  - (we also have a VTK translation layer)
- Steps towards productization
  - flesh out iterators, algorithms
  - spend more time on optimization
  - improve client-facing API
- First release available at github
  - https://github.com/jsmeredith/EAVL
  - http://ft.ornl.gov/eavl/
- Papers:
  - J.S. Meredith, R. Sisneros, D. Pugmire, S. Ahern, "**A Distributed Data-Parallel Framework for Analysis and Visualization Algorithm Development**" , Fifth Workshop on General Purpose Processing on Graphics Processing Units (GPGPU5), 2012.
  - J.S. Meredith, S. Ahern, D. Pugmire, R. Sisneros, "**EAVL: The Extreme-scale Analysis and Visualization Library"**, Eurographics Symposium on Parallel Graphics and Visualization (EGPGV) in association with Eurographics, 2012.
- SC12 Panel on next generation vis/analysis frameworks