Data Wrangling with the SheafSystem™

An Open Source Tool for Scientific Data Management

David M. Butler, President
Limit Point Systems, Inc.
d.m.butler@limitpoint.com
www.limitpoint.com

Data wrangling in scientific computing

- "data wrangling" refers to the chores required to interoperate diverse data formats in scientific computing workflows
- tools have not received same degree of attention or level of technical sophistication as the central processes (PDE solvers, etc)
- dearth of general, high-level, easy-to-use tools limits both programmer and user productivity

➔similar problem in business computing

Data wrangling in business computing

- SQL/relational data model provides
  - general conceptual framework
  - rich tool ecosystem
- NoSQL data models extend ecosystem to less structured data
- none of these data models fit scientific computing well
  - data is *mathematically* structured
  - more structured, not less structured

➔scientific computing needs a general data model

Data models for scientific computing

- many attempts over last several decades to find general data model for scientific computing
- one approach: leverage the fact that scientific computing is already mathematized
  - don't need to invent new abstractions
  - science has spent centuries developing them
  - we just need to use them

➔fiber bundle data model based on this approach

Fiber bundle data model

- taken directly from mathematical physics
- objects
  - topological spaces (meshes)
  - multilinear spaces (scalar, vector, tensor, ...)
  - sections of fiber bundles (fields)
- used successfully in a number of projects
  - IBM Data Explorer (now OpenDX)
  - F5 library
- provides good description of how we want to use data

→experience exposes limitations

Limitations of fiber bundle model

- doesn't describe how the objects are represented as data
- representation is description of how high-level semantic objects are decomposed into low-level primitives
- fiber bundle model simply doesn't address this
- implementation requires ad-hoc extensions

➔ad-hoc extensions limit scope of the implementation

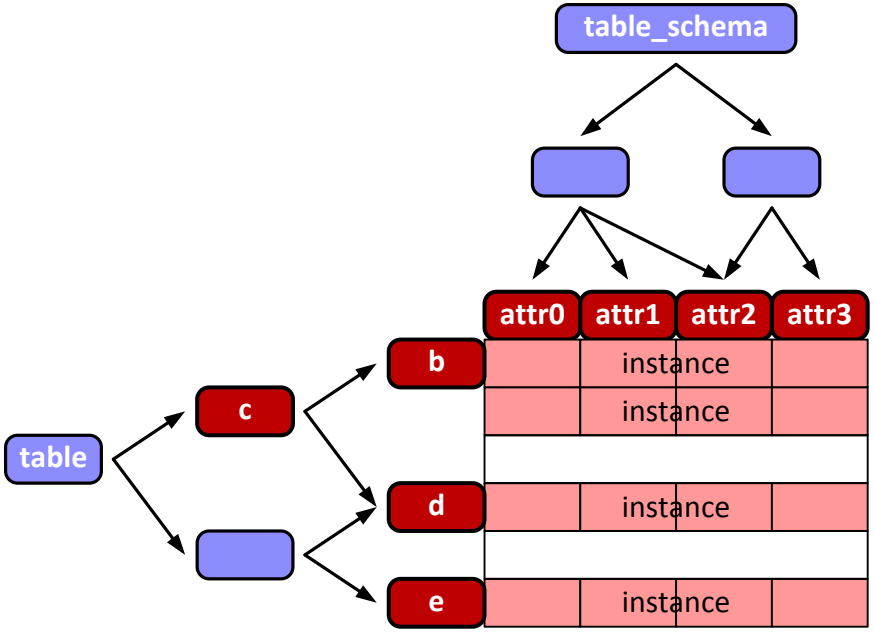
Sheaf data model

- addresses decomposition at fundamental mathematical level
- objects
  - sheaves of lattice-ordered relations
  - details: http://www.limitpoint.com/index.php/publications
- informally
  - finite distributive lattice describes decomposition into parts
  - sheaf describes association of attributes with parts

➔can be visualized using table metaphor

Table metaphor

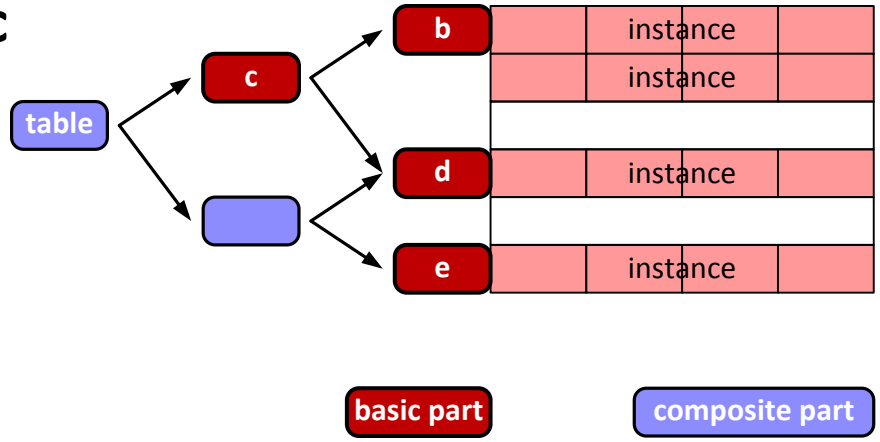
- similar to relational model table metaphor
- database is a collection of tables
- each table represents a type
- each row represents an instance of the type
- each column represents an attribute of the type



➔rows and columns are lattice-ordered

Rows equipped with client-defined lattice order

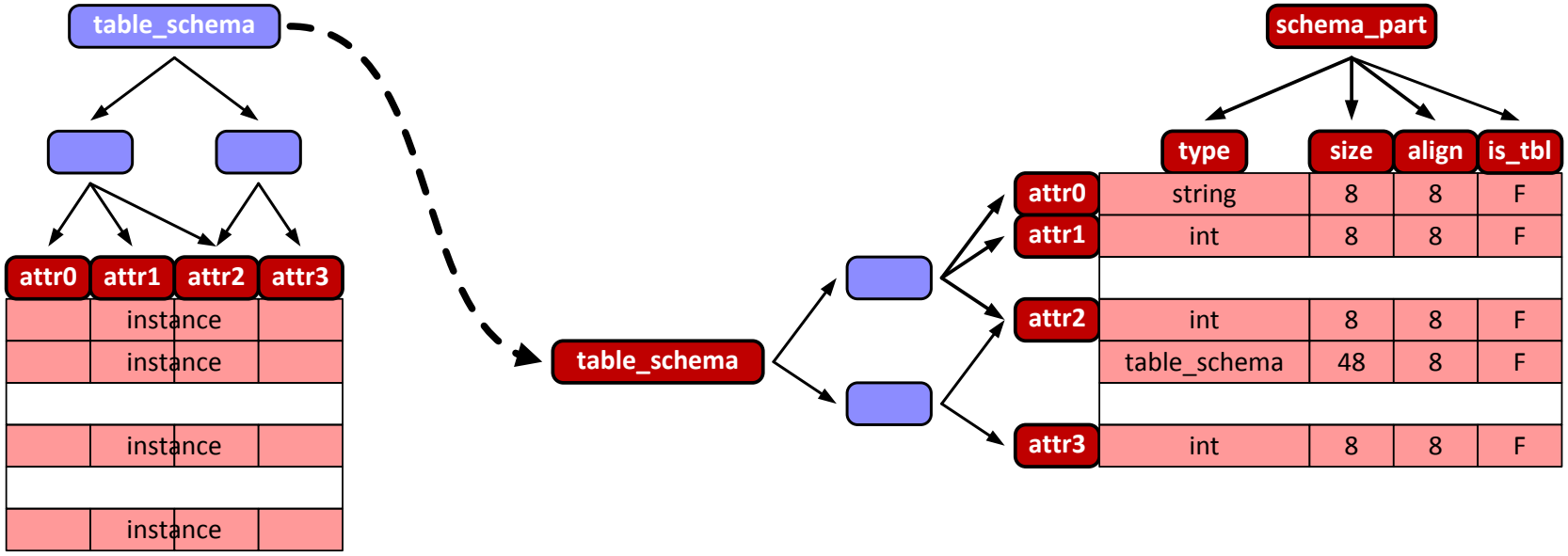
- finite distributive lattice (FDL)
  - all distinct composite parts
  - built from given basic parts
  - with inclusion relationships
- visualize as directed acyclic graph
  - basic part nodes
  - composite part nodes
  - links = direct inclusion
- row for each basic part



➔schema for table is also lattice ordered

Schema for table is lattice ordered

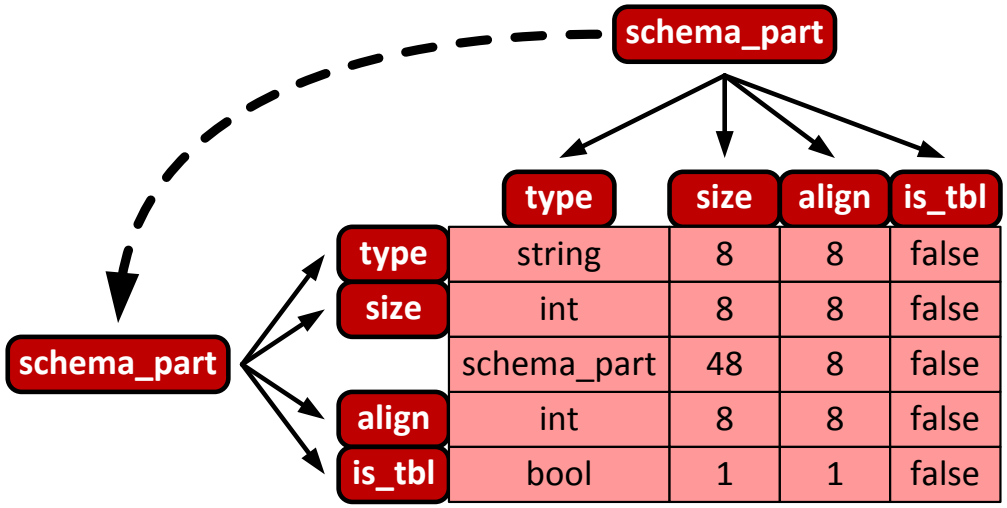
- column lattice specified by row lattice of some other table



➔each table must have another table as its schema
➔where does recursion terminate?

Recursion terminates in *schema_part*

- schema for *schema_part* table is its own row lattice



➔schemas are first class data objects
➔dependent type system

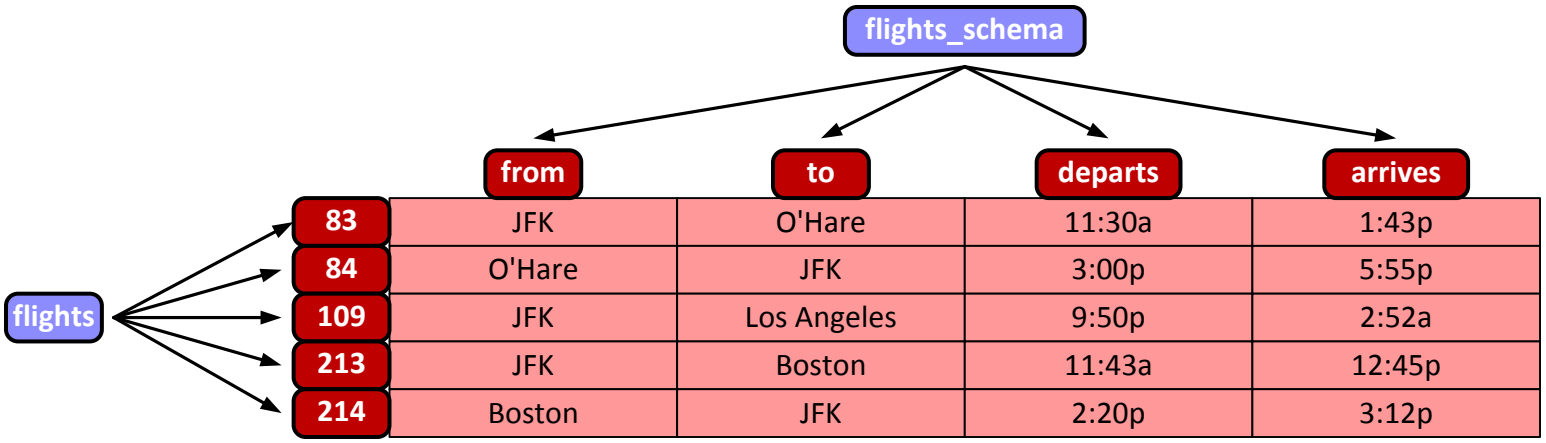
Dependent type system

- Can rigorously represent all the important types of scientific computing in a single unified formalism
  - tabular types
  - spatial types
  - physical property types
  - section types

➔examine each in more detail

Tabular types

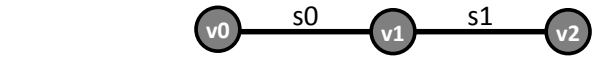
- conventional relational tables
- rows and columns unordered
- corresponds to Boolean lattice
- basic parts with no links between them



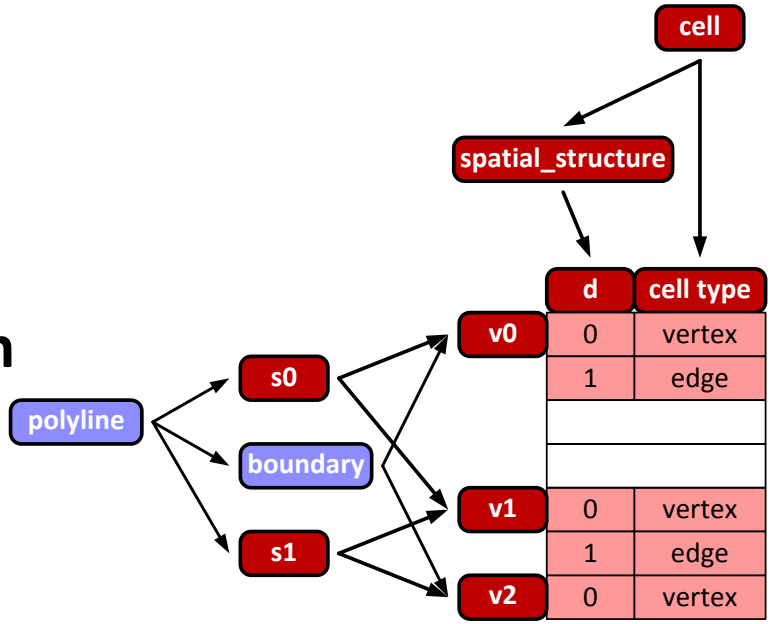
➔relational model is limiting case of sheaf model

Spatial types (meshes)

- mesh is a decomposition of space into basic parts
  - zones, faces, edges, vertices
- any decomposition of space can be represented as an FDL
  - nodes represent parts of space
  - links represent spatial inclusion
- rigorous connection between data and math semantics
  - cell complexes
  - topological spaces



1D mesh (polyline) with 2 segments



sheaf table for polyline

➔FDL is an algebraic representation

FDL is an algebraic representation

- supports broad set of spatial operations
  - union (“join”)
  - intersection (“meet”)
  - sum
  - difference
  - Cartesian product
  - tensor product

➔basis for powerful query language

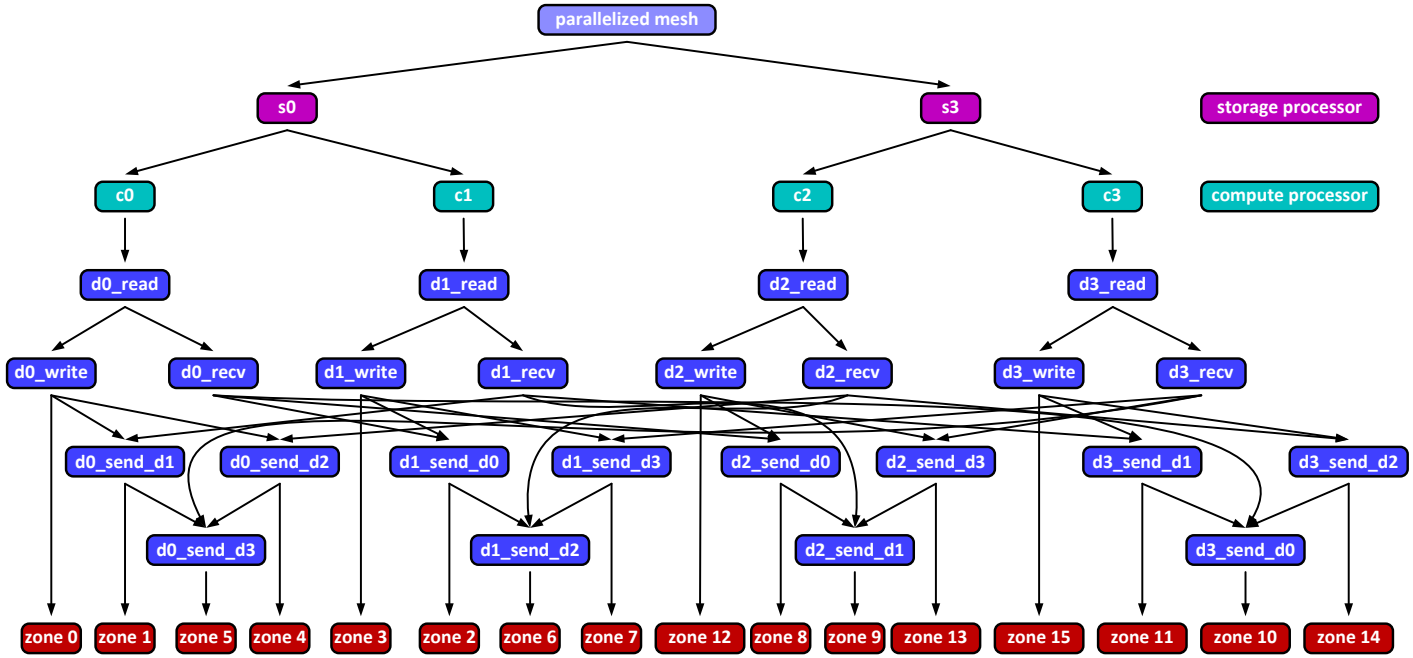
FDL supports multiple concurrent decompositions ("MCDs")

- mesh defines basic parts
- can form composite parts for any purpose
  - domains for parallel processing
  - parts for design and manufacture
  - level sets of some property value for analysis
  - geological, geographical, and political regions
  - any ad-hoc client-defined grouping
- any and all combinations can be represented concurrently
- any and all combinations can be created at run-time

➔no need for specific decompositions to be "designed-in"

FDL supports parallelism

- mesh and multicomputer each described by FDL
- distribution described by order preserving map
- map pulls computer FDL back into mesh FDL

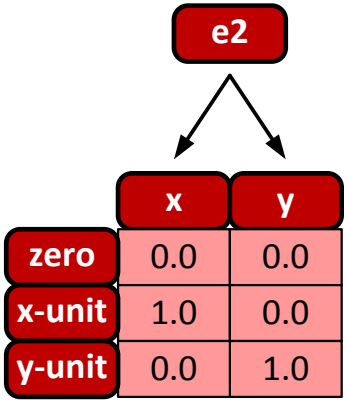


➔unified description of mesh and computation resources

➔framework for automated distribution and parallelism services

Physical property types (general object-oriented types)

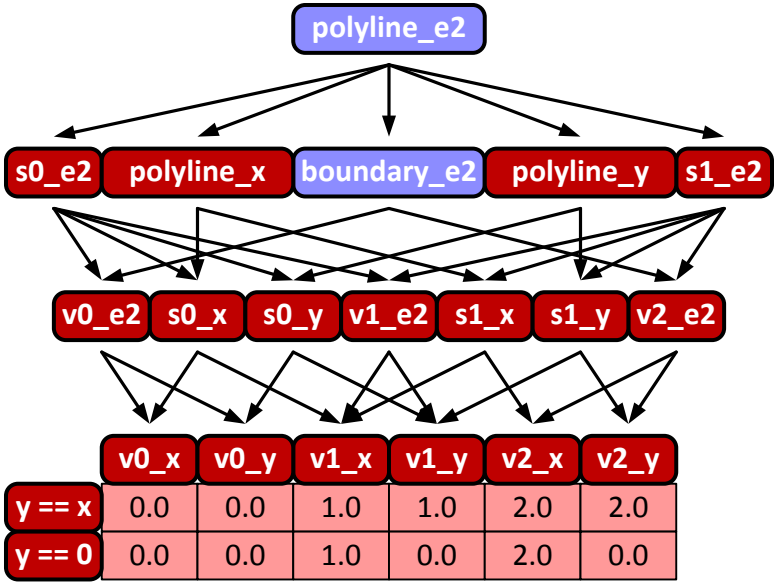
- rows typically unordered
- column order represents subobject inclusion
  - aggregation (data members)
  - inheritance (base classes)
- strong typing as opposed to loose collections of attributes
- rigorous connection between data and math semantics
- first class objects support rigorous sharing of type specs



sheaf table for Euclidean vector space of dimension 2 (simplified schema)

Section types (property vs position)

- rigorous notion of schema
- algebraic construction
  - row lattice of mesh
  - column lattice of property
- typically lattice tensor product
  - (mesh row) ⊗ (property col)
- schema describes section data concretely and completely
- rigorous connection between data and math semantics



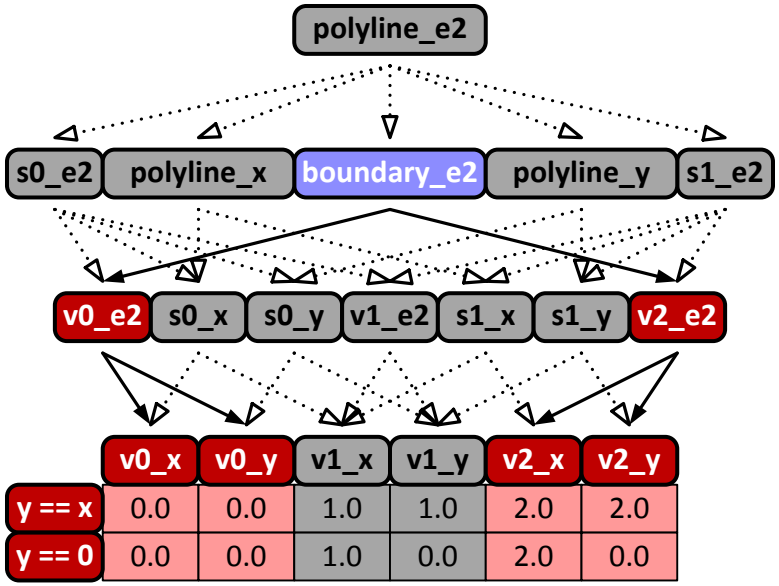
	v0_x	v0_y	v1_x	v1_y	v2_x	v2_y
y == x	0.0	0.0	1.0	1.0	2.0	2.0
y == 0	0.0	0.0	1.0	0.0	2.0	0.0

sheaf table for e2 on polyline
with 2 sections ("y==x" and "y==0")

➔supports restriction

Section schema supports restriction

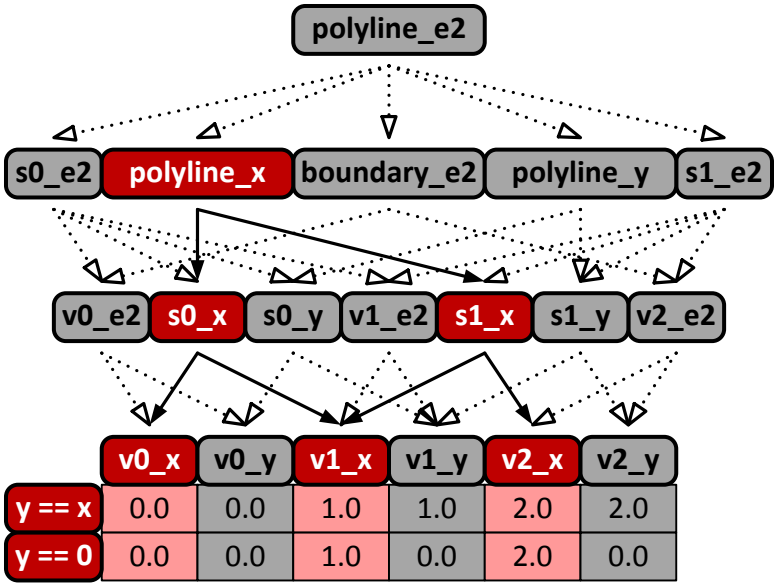
- to any part of mesh



	v0_x	v0_y	v1_x	v1_y	v2_x	v2_y
y == x	0.0	0.0	1.0	1.0	2.0	2.0
y == 0	0.0	0.0	1.0	0.0	2.0	0.0

sheaf table for e2 on polyline restricted to boundary

- to any part of property



	v0_x	v0_y	v1_x	v1_y	v2_x	v2_y
y == x	0.0	0.0	1.0	1.0	2.0	2.0
y == 0	0.0	0.0	1.0	0.0	2.0	0.0

sheaf table for e2 on polyline restricted to x component

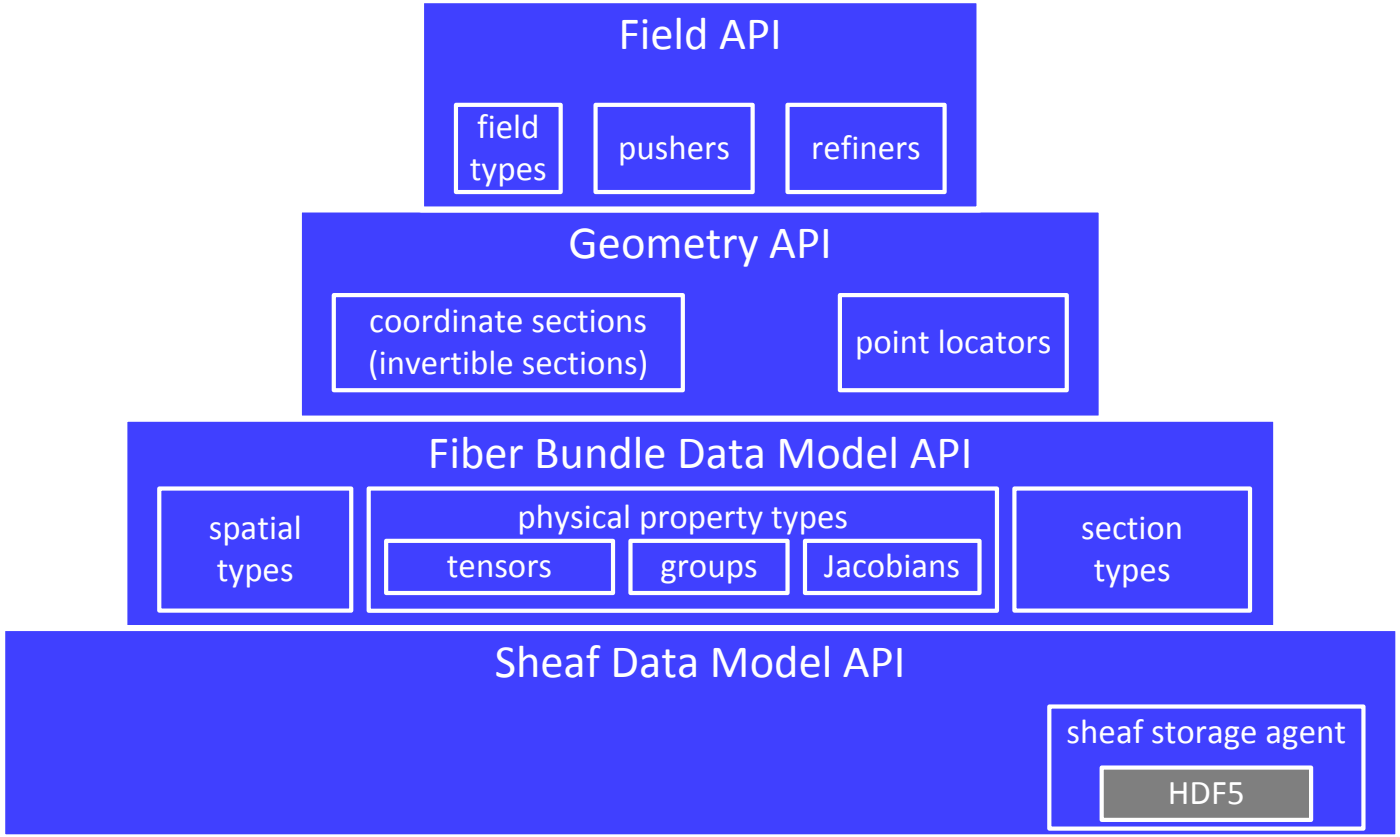
➔ important for parallel processing, visualization

Fiber bundle model revisited

- fiber bundle model can be implemented on top of sheaf
- layered abstractions provide
  - high-level operations of fiber bundle model
  - complete and detailed description of data
- any mesh type
- any property type
- any interpolation method

➔addresses entire scope of scientific computing

SheafSystem™ architecture



➔implemented in C++ with Python bindings

Sheaf API

- create and delete tables and schema
- graph facet
  - create and delete basic and composite parts
  - create and delete cover links
- algebraic facet
  - find or create join and meet of parts
- attribute facet
  - put and get attributes of part
- query facet
  - iterate over lower or upper cover of part
  - iterate over downset or upset of part

→ops for defining, storing, accessing types and instances

→~245 classes, 224K lines of code

Fiber bundle API

- uses Sheaf API to define types for scientific computing
- spatial types facet
  - representation of arbitrary spatial decompositions
  - especially cell complexes
- property types facet
  - common algebraic types of mathematical physics
- section types facet
  - sections of fiber bundles
  - property value as function of (cell, local coordinates)

➔~308 classes, 370K lines of code total

➔examine each facet in more detail

Spatial types facet

- manipulate, store arbitrary spatial decompositions
  - point meshes
  - structured meshes (grids)
  - unstructured meshes (triangle, quad, tet, hex, etc)
  - multi-block with shared parts
  - client-defined high-level MCDs
  - any other spatial decomposition
- algebraic operations
  - meet (intersection) and join (union)
  - Cartesian and tensor products (work in progress)

➔~40 classes, 44K lines of code
➔client extensible

Property types

- tensor hierarchy
  - abstract vectors Vd
  - general tensors Tp
  - antisymmetric tensors ATp
  - symmetric tensors STp
  - Euclidean vectors Ed
  - p = 0, 1, 2, 3, 4
  - d = 1,2,3,4
- transformation groups
  - gln, n = 1,2,3
- Jacobians

- property algebra
  - scalar algebra
  - abstract vector algebra
  - general tensor algebra
  - exterior algebra
  - symmetric algebra
  - Euclidean vector algebra
  - coordinate transformations

➔~60 classes, 113K lines of code
➔client extensible

Section types

- any mesh
- any property
  - section types for all library defined property types
- any interpolation
  - interpolators for all library defined property and cell types
- section algebra
  - all property operations propagated to sections
- section restriction
  - restrict to any part of mesh
  - restrict to any part of property

➔~100 classes, 100K lines of code
➔client extensible

Geometry API

- adds notion of invertible section to fiber bundles API
- (cell, local coordinates) as function of property value
- used to represent global coordinates section
- supports point location queries

➔~30 classes, 22K lines of code

Field API

- field is pair (global coordinates section, property section)
  - defined on same mesh
- field types for all library defined property types
- operations
  - property value as function of global coordinates value
  - field algebra
  - field calculus
  - moving properties between meshes (“remapping”)
  - property controlled mesh refinement

➔~100 classes, 77K lines of code

Data wrangling with the SheafSystem™

- SheafSystem™ provides high-level "loop-free" operations for data wrangling in scientific computing
  - mesh construction, manipulation, and refinement
  - section and field restriction and extension
  - section and field algebra
  - field calculus
  - remapping
- mathematically rigorous and complete
  - any type of mesh
  - any type of property
  - any type of interpolation
- data persistence with complete mathematical semantics

➔Python is natural binding for this functionality

Status and technology roadmap

- **status**
  - **in commercial use now at workstation scope**
- **technology roadmap**
  - **3 year plan to scale up to enterprise scope**

Development track	Objective
Interoperation services	Adapters for wide set of external formats Additional functionality
Geometry services	Native geometry engine
Parallelism services	Automatic decomposition and distribution
Database services	Scalable distributed storage Sheaf query language

Open SheafSystem™

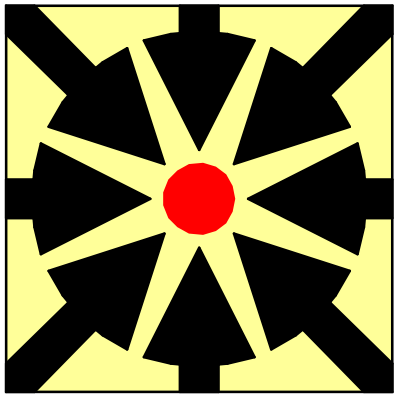
- currently preparing open source release
- expected release date July 2013
  - just as soon as we figure out licensing!
- tentatively planning to use Apache license
  - want to encourage widest possible use
  - by both non-profit and commercial organizations
- need your input!
  - does Apache license work for you?
  - does it create any barriers to usage?

➔contact me at SciPy by calling 925-243-8891
➔or email d.m.butler@limitpoint.com with comments

Acknowledgements

- Original research and development funded by subcontracts B347785, B515090, and B560973 of prime contract W-7405-ENG-48 with the Department of Energy National Nuclear Security Administration (DOE/NNSA)

- Ongoing development has been funded by Shell GameChanger and Shell TaCIT



END