HDF5 FastQuery

Accelerating Complex Queries on HDF Datasets using Fast Bitmap Indices

John Shalf, Wes Bethel

LBNL Visualization Group

Kensheng Wu, Kurt Stockinger

LBNL SDM Center

Luke Gosink, Ken Joy

UC Davis IDAV

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

Motivation and Problem Statement

- Too much data.
- Visualization “meat grinders” not especially responsive to needs of scientific research community.
- What scientific users want:
  - Scientific Insight
  - Quantitative results
  - Feature detection, tracking, characterization
  - (lots of bullets here omitted)
- See:
  - http://vis.lbl.gov/Publications/2002/VisGreenFindings-LBNL-51699.pdf
  - http://www-user.slac.stanford.edu/rmount/dm-workshop-04/Final-report.pdf

Motivation and Problem Statement

- Too much data.
- Visualization “meat grinders” not especially responsive to needs of scientific research community.
- What scientific users want:
  - Scientific Insight
  - Quantitative results
  - Feature detection, tracking, characterization
  - (lots of bullets here omitted)
- See:
  - http://vis.lbl.gov/Publications/2002/VisGreenFindings-LBNL-51699.pdf
  - http://www-user.slac.stanford.edu/rmount/dm-workshop-04/Final-report.pdf

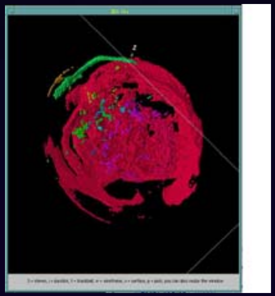
What is FastBit?

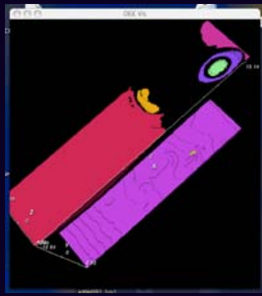
(what is it's role in data analysis?)

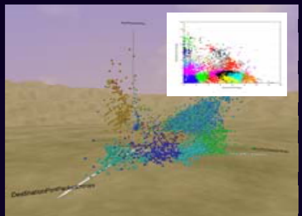
Using Indexing Technology to Accelerate Data Analysis

- Use cases for indexed datasets
  - Support Compound Range Queries: eg. *Get me all cells where Temperature > 300k AND Pressure is < 200 millibars*
  - Subsetting: Only load data that corresponds to the query.
    - Get rid of visual “clutter”
    - Reduce load on data analysis pipeline
  - Quickly find and label connected regions
  - Do it really fast!

- Applications
  - Astrophysics:
    - Remove clutter from messy supernova explosions
  - Combustion:
    - Locate and track ignition kernels
  - Particle Accelerator Modeling:
    - identify and select errant electrons
  - Network Security Data:
    - Pose queries against enormous packet logs
    - Identify candidate security events

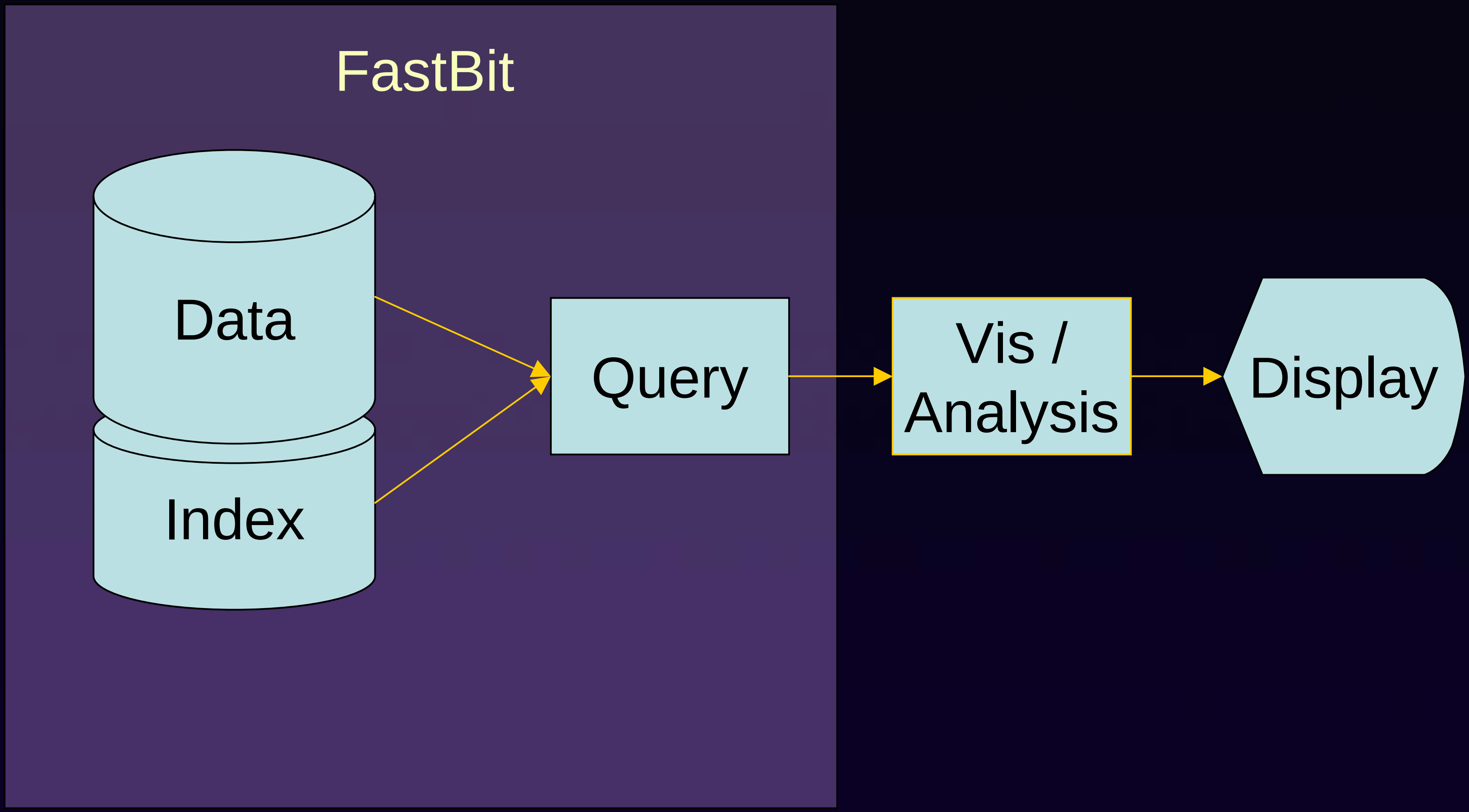






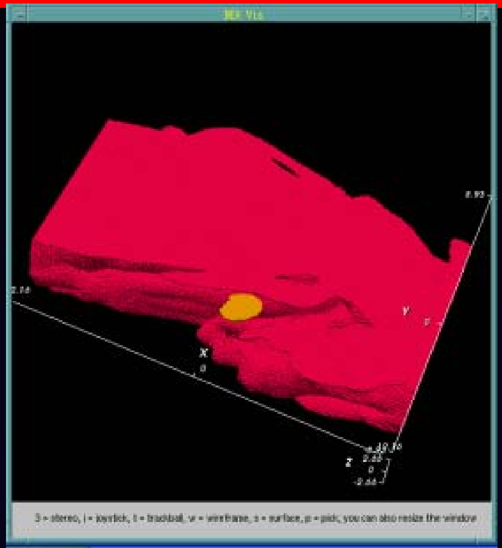
Architecture Overview: Generic Visualization Pipeline

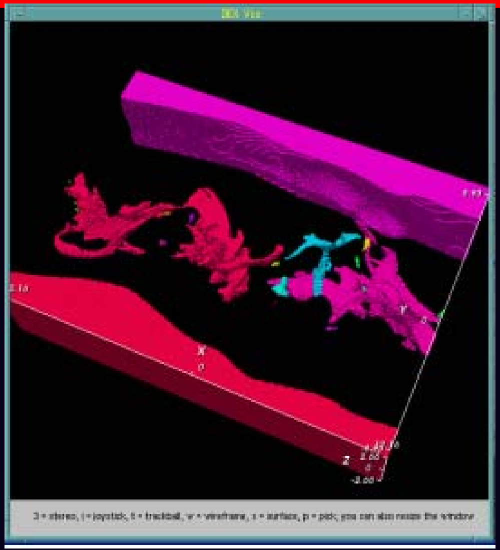
Architecture Overview: Query-Driven Vis. Pipeline



Query-Driven Subsetting of Combustion Data Set

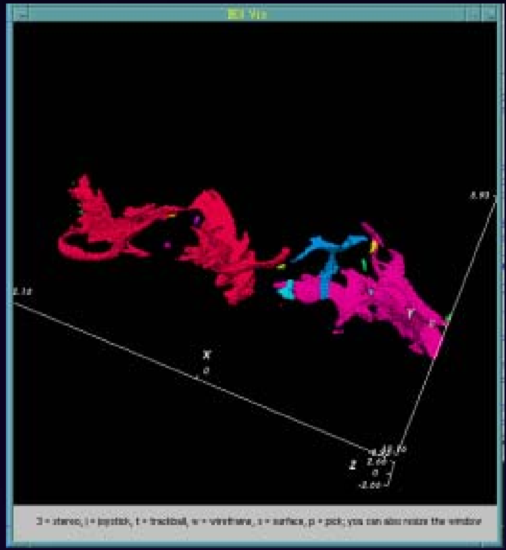
a) Query: CH4 > 0.3

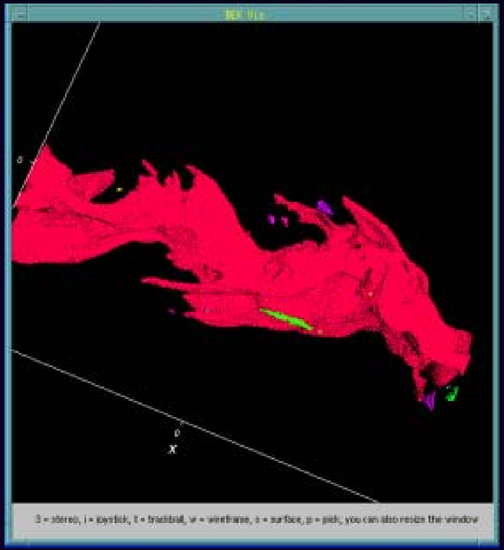




b) Q: temp < 3

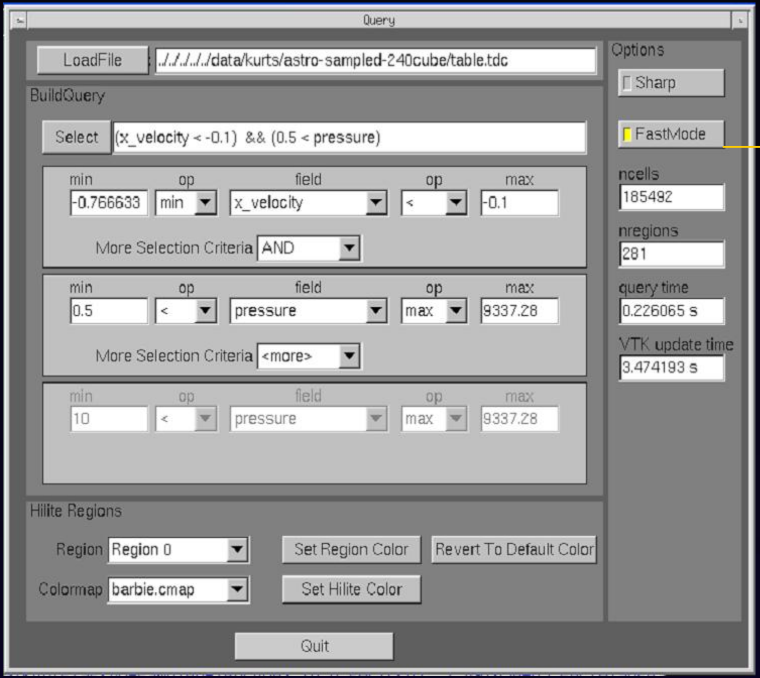
c) Q: CH4 > 0.3 AND temp < 3

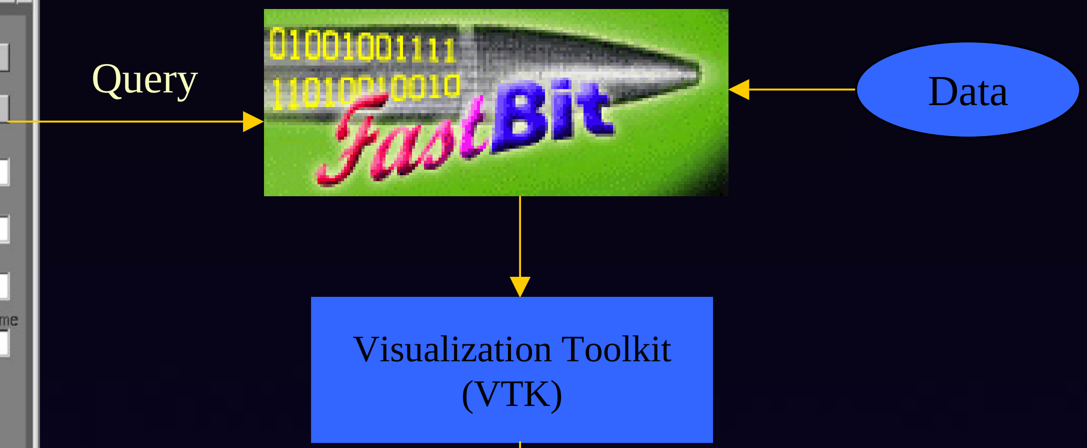


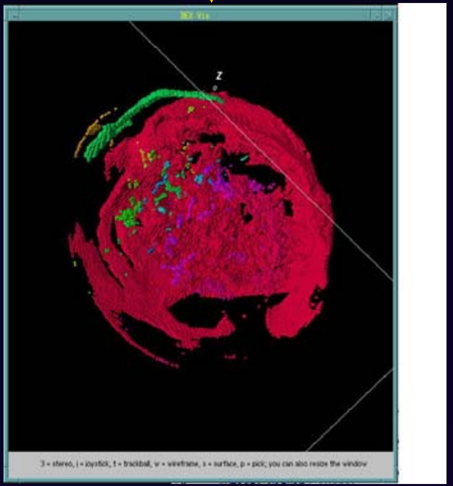


d) Q: CH4 > 0.3 AND temp < 4

DEX Visualization Pipeline

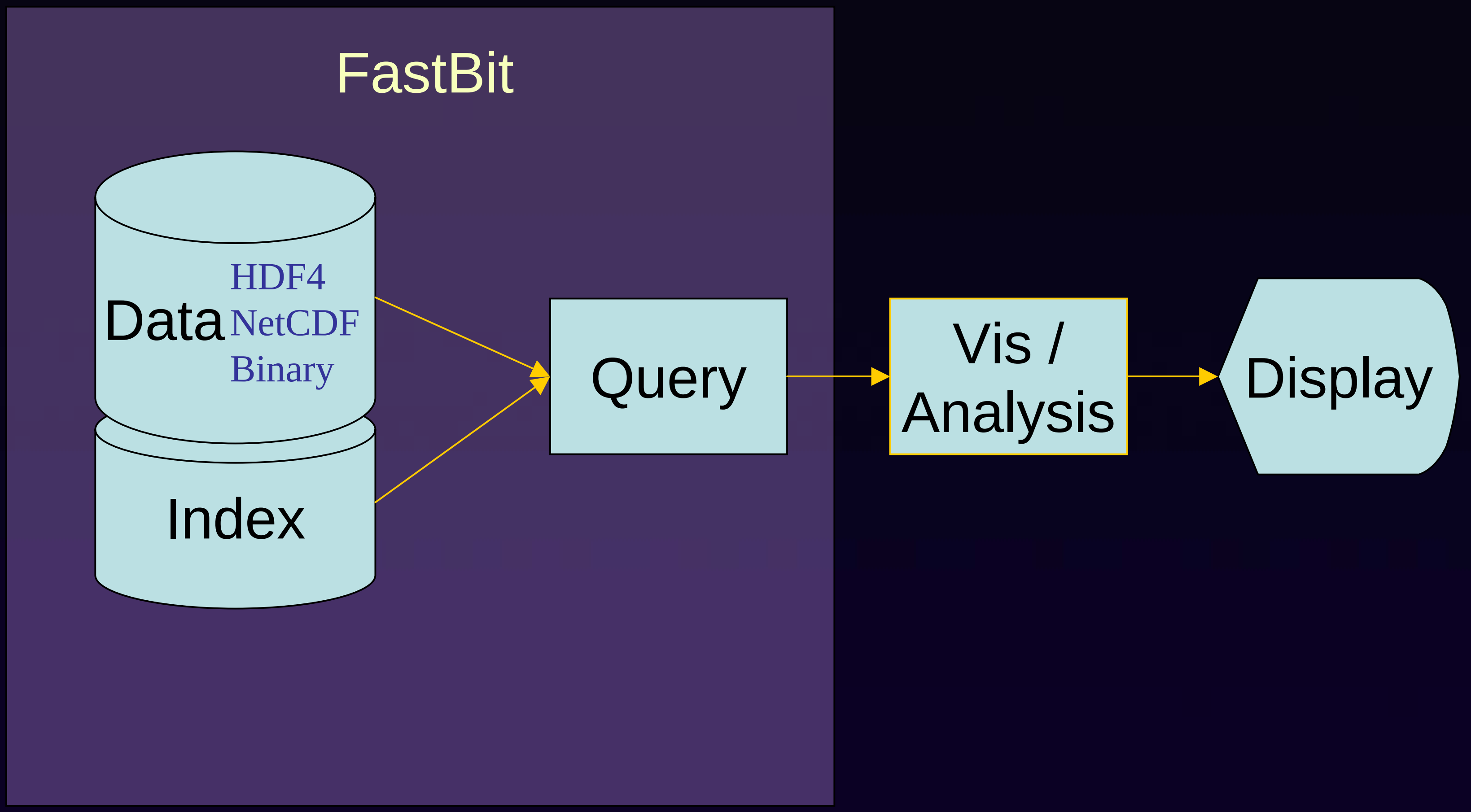




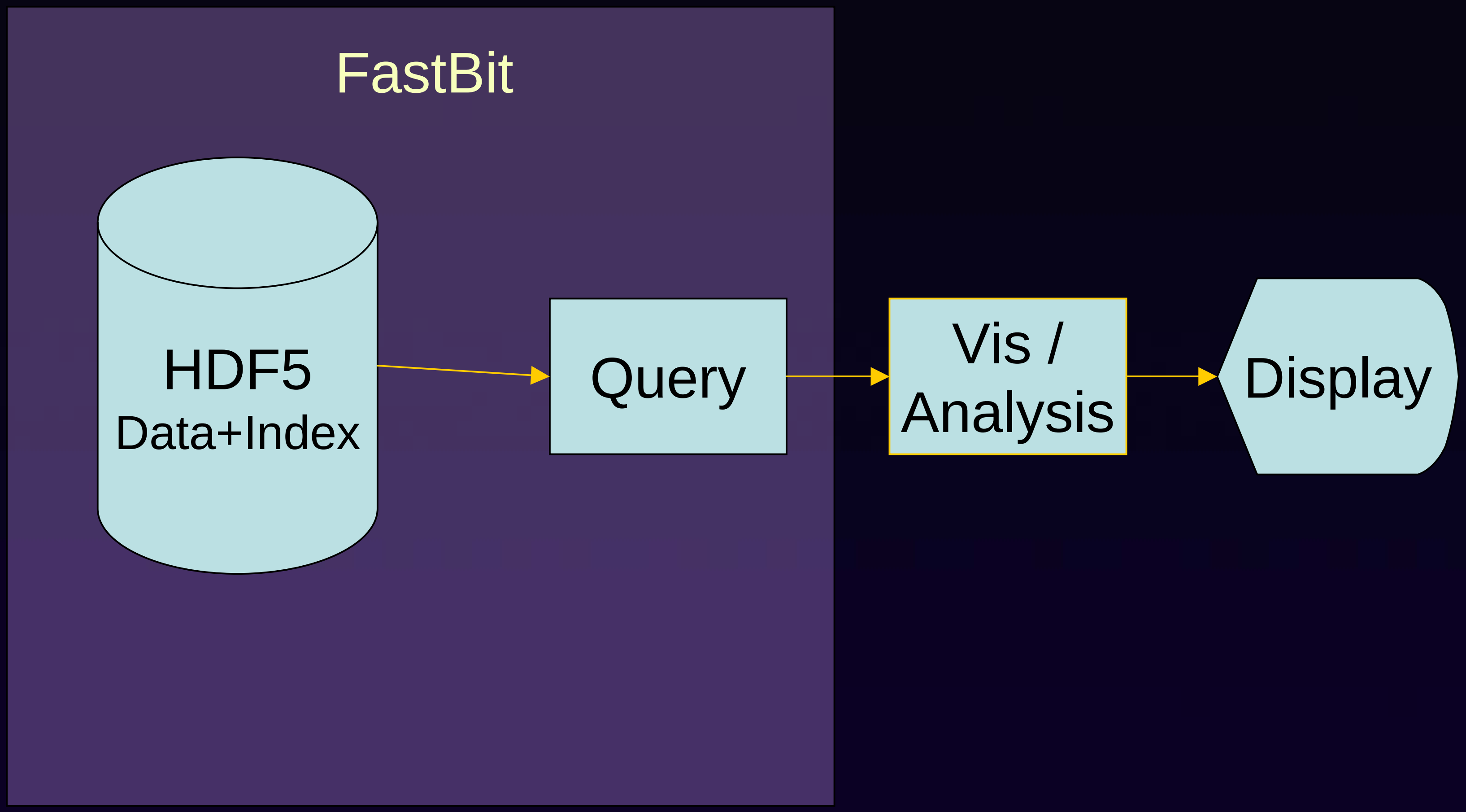


3D visualization of a Supernova explosion

Architecture Overview: Query-Driven Analysis Pipeline



Architecture Overview: Query-Driven Analysis Pipeline



How do Fast Bitmap Indices Work?

Why Bitmap Indices?

- Goal: efficient search of *multi-dimensional* read-only (append-only) data:
  - E.g. temp < 104.5 AND velocity > 10^7 AND density < 45.6

- Commonly-used indices are designed to be updated quickly
  - E.g. family of B-Trees
  - Sacrifice search efficiency to permit dynamic update

- Most multi-dimensional indices suffer *curse of dimensionality*
  - E.g. R-tree, Quad-trees, KD-trees, …
  - Don't scale to large number of dimensions (< 10)
  - Are efficient only if all dimensions are queried

- Bitmap indices
  - Sacrifice update efficiency to gain more search efficiency
  - Are efficient for multi-dimensional queries
  - Query response time scales linearly in the actual number of dimensions in the query

What is a Bitmap Index?

Data values	b_0	b_1	b_2	b_3	b_4	b_5
0	1	0	0	0	0	0
1	0	1	0	0	0	0
5	0	0	0	0	0	1
3	0	0	0	1	0	0
1	0	1	0	0	0	0
2	0	0	1	0	0	0
0	1	0	0	0	0	0
4	0	0	0	0	1	0
1	0	1	0	0	0	0

- Compact: one bit per distinct value per object.
- Easy and fast to build: O(*n*) vs. O(*n* log *n*) for trees.
- Efficient to query: use bitwise logical operations.
  - (0.0 < H_2O < 0.1) AND (1000 < temp < 2000)
- Efficient for multidimensional queries.
  - No "curse of dimensionality"
- What about floating-point data?
  - Binning strategies.

Bitmap Index Encoding

List of Attributes | Equality Encoding | Range Encoding

	$\pi_A(R)$	E^9	E^8	E^7	E^6	E^5	E^4	E^3	E^2	E^1	E^0	R^8	R^7	R^6	R^5	R^4	R^3	R^2	R^1	R^0
1	3	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	0	0	0
2	2	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	0	0
3	1	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	0
4	2	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	0	0
5	8	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
6	2	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	0	0
7	9	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
9	7	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
10	5	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0
11	6	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
12	4	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	0	0	0	0

(a) (b) (c)

- Equality encoding compresses very well
- Range encoding optimized for one-sided range queries, e.g. temp < 3

Performance

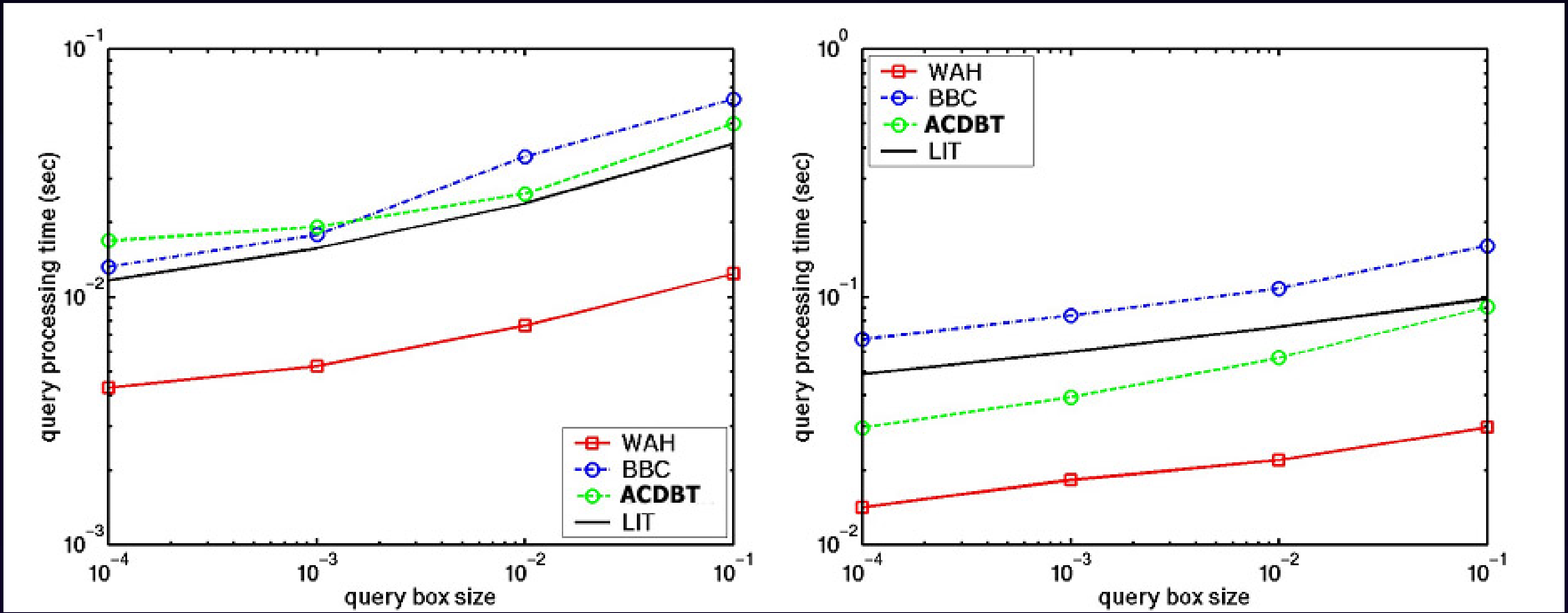
Bitmap Index Query Complexity and Space Requirements

- How Fast are Queries Answered?
  - Let N denote the number of objects and H denote the number of hits of a condition.
  - Using uncompressed bitmap indices, search time is O(N)
  - With a good compression scheme, the search time is O(H) – the theoretical optimum

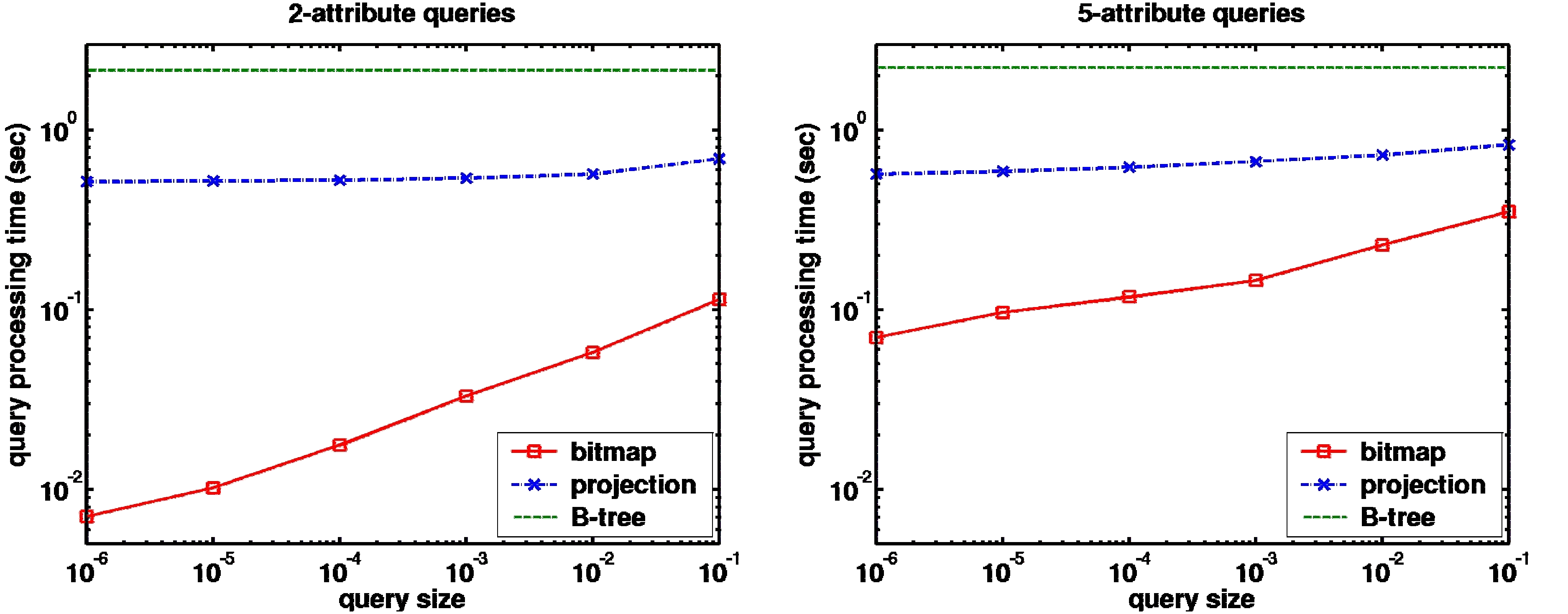
- How Big are the Indices?
  - In the worst case (completely random data), the bitmap index requires about 2x in data size for one variable (typically 0.3x).
  - In contrast, 4x space requirement not uncommon for tree-based methods for one variable.
  - Curse of dimensionality: for N points in D dimensions:
    - Bitmap index size: O(D*N)
    - Tree-based method: O(N**D)!!!

Compressed Bitmap Index Query Performance

- FastBit Word-Aligned Hybrid (WAH) compression performance better than commercial systems.
- Different bitmap compression technologies have different performance characteristics.

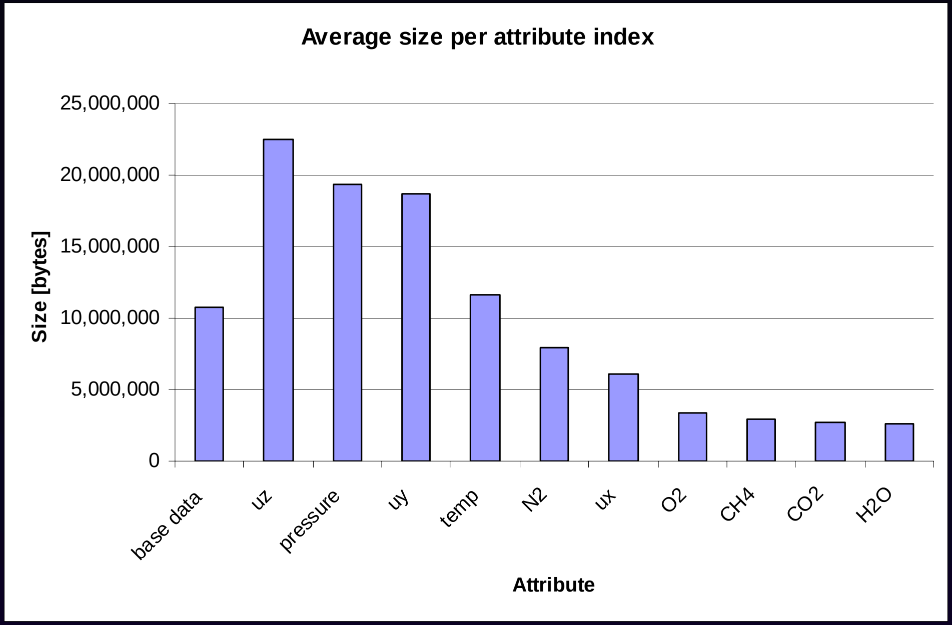


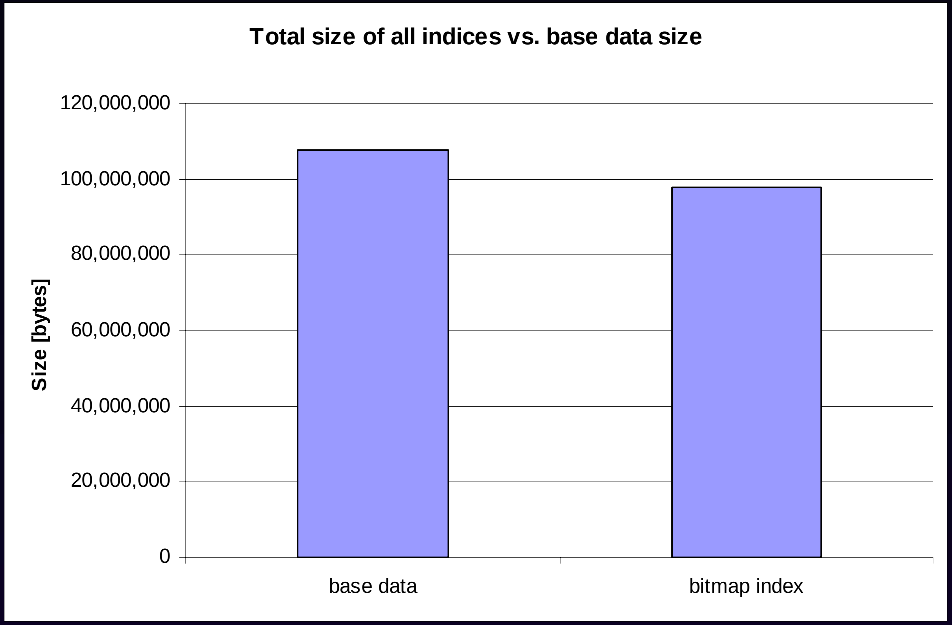
Queries Using Bitmap Indices are Fast



- Log-log plot of query processing time for different size queries
- The compressed bitmap index is at least 10X faster than B-tree and 3X faster than the projection index

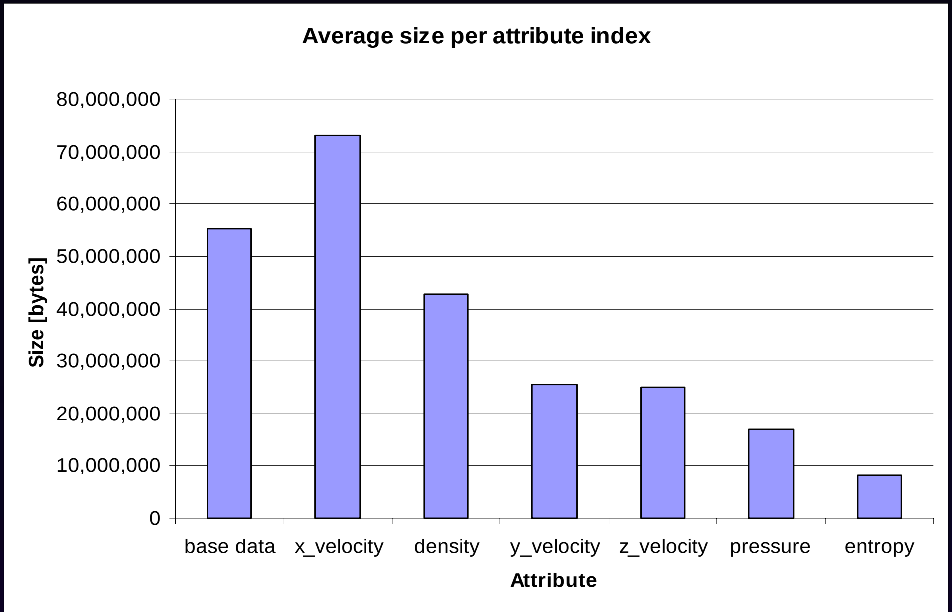
Size of Bitmap Index vs. Base Data *(Combustion)*

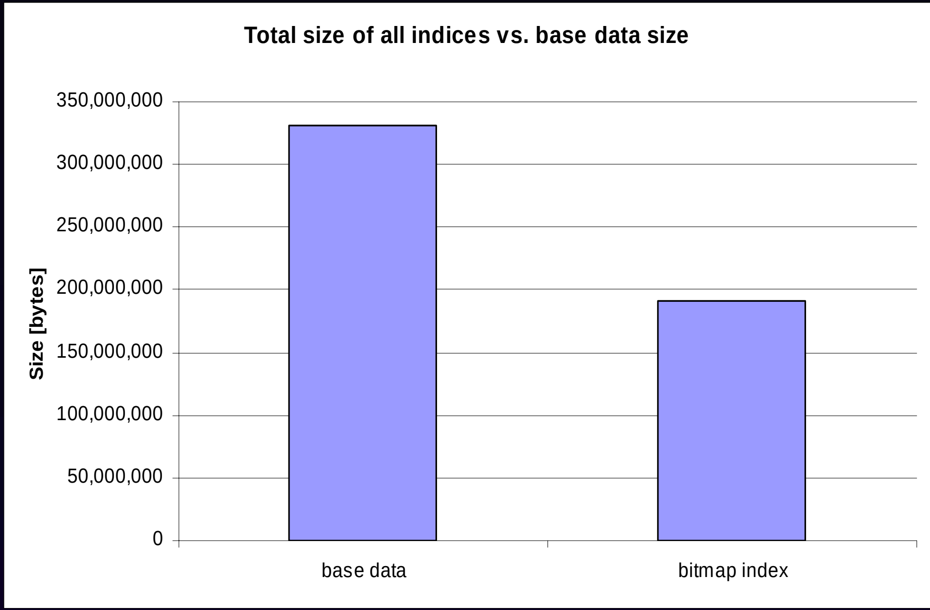




- Compressed bitmap index with 100 range-encoded bins is about same size as base data.
- Note: B-tree index is about 3 times the size of the base data.
- Building the index takes ~5 seconds for 100Megs on P4 2.4GHz workstation

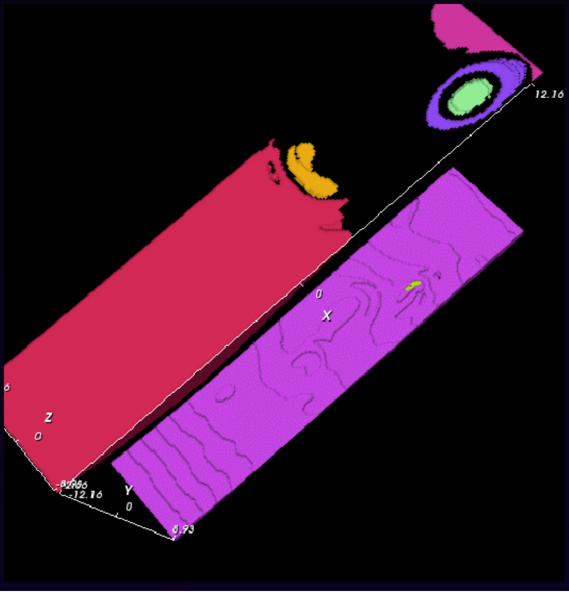
Size of Bitmap Index vs. Base Data (Astrophysics)

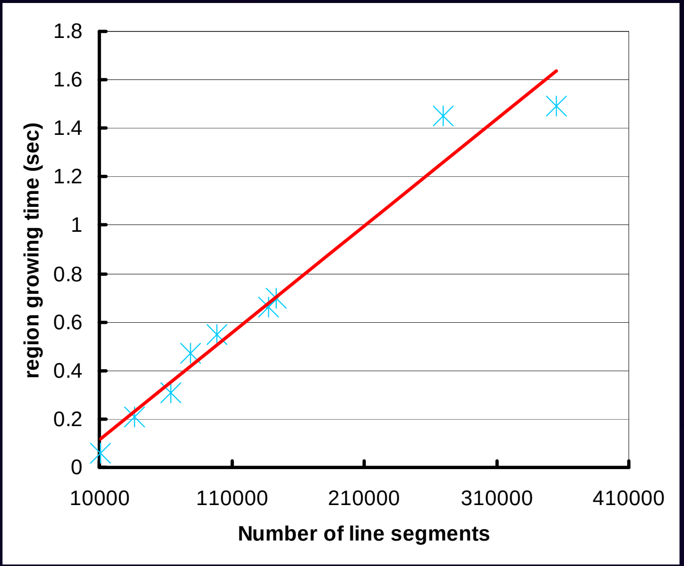




- Size of compressed bitmap index is only 57% of base data.
- Building an index for all attributes takes ~17 seconds for 340 Megs.

Region Growing and Connected Component Labeling





- The result of the bitmap index query is a set of blocks.
- Given a set of blocks, find connected regions and label them.

- Region growing scales linearly with the number of cells selected.

HDF5 - FastQuery File Organization

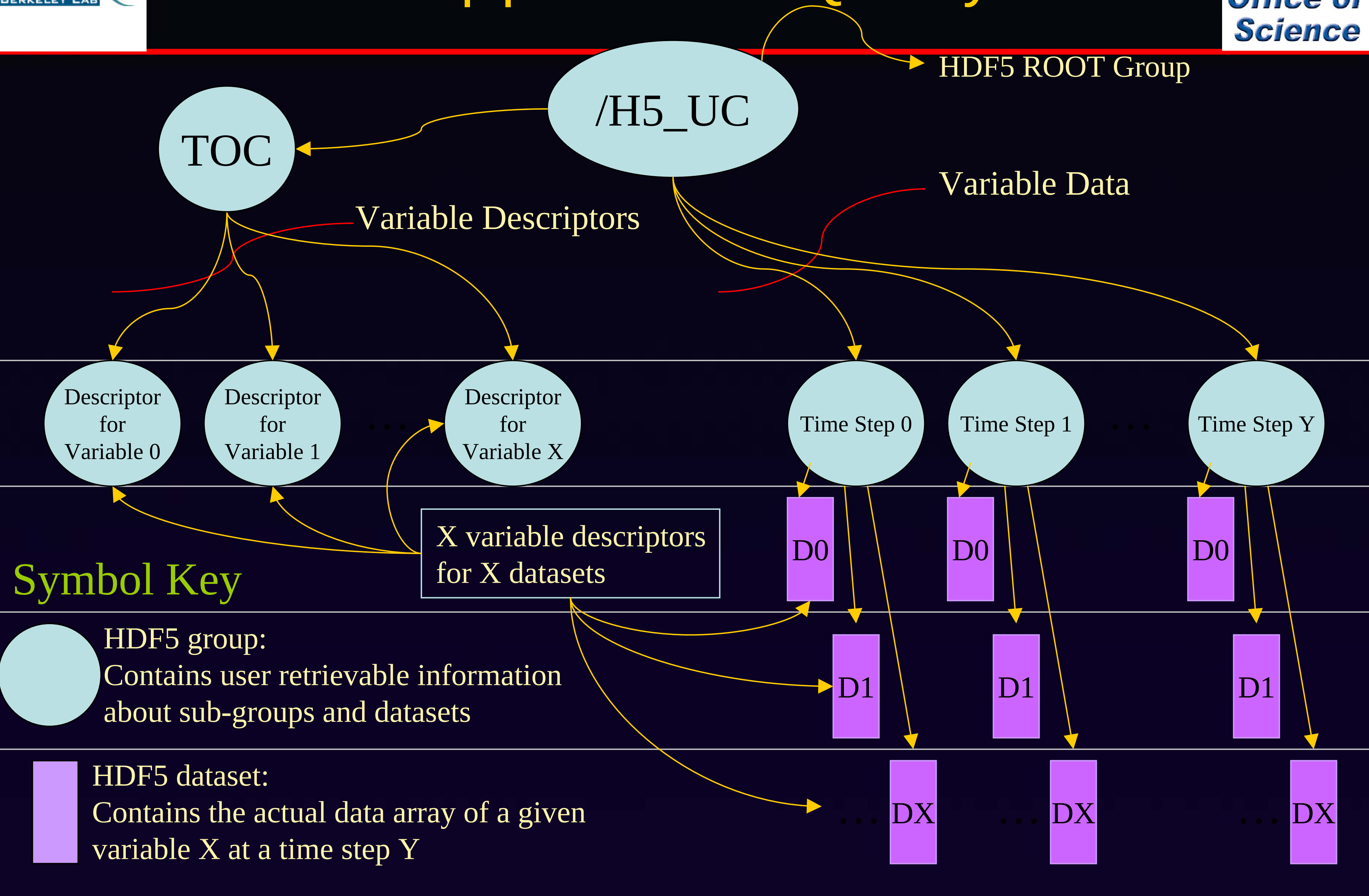
File Organization

- Current
  - Data in HDF4, NetCDF converted to raw binary
  - One file per species + one file per index
  - ASCII file for metadata
  - One directory per timestep
  - Non-portable binary (must byte-swap data)
- HDF5 FastQuery
  - Indices + data all in same file
  - Machine independent binary representation
  - Multiple time-steps per file
  - Pose queries against data stored in “indexed” HDF5 file

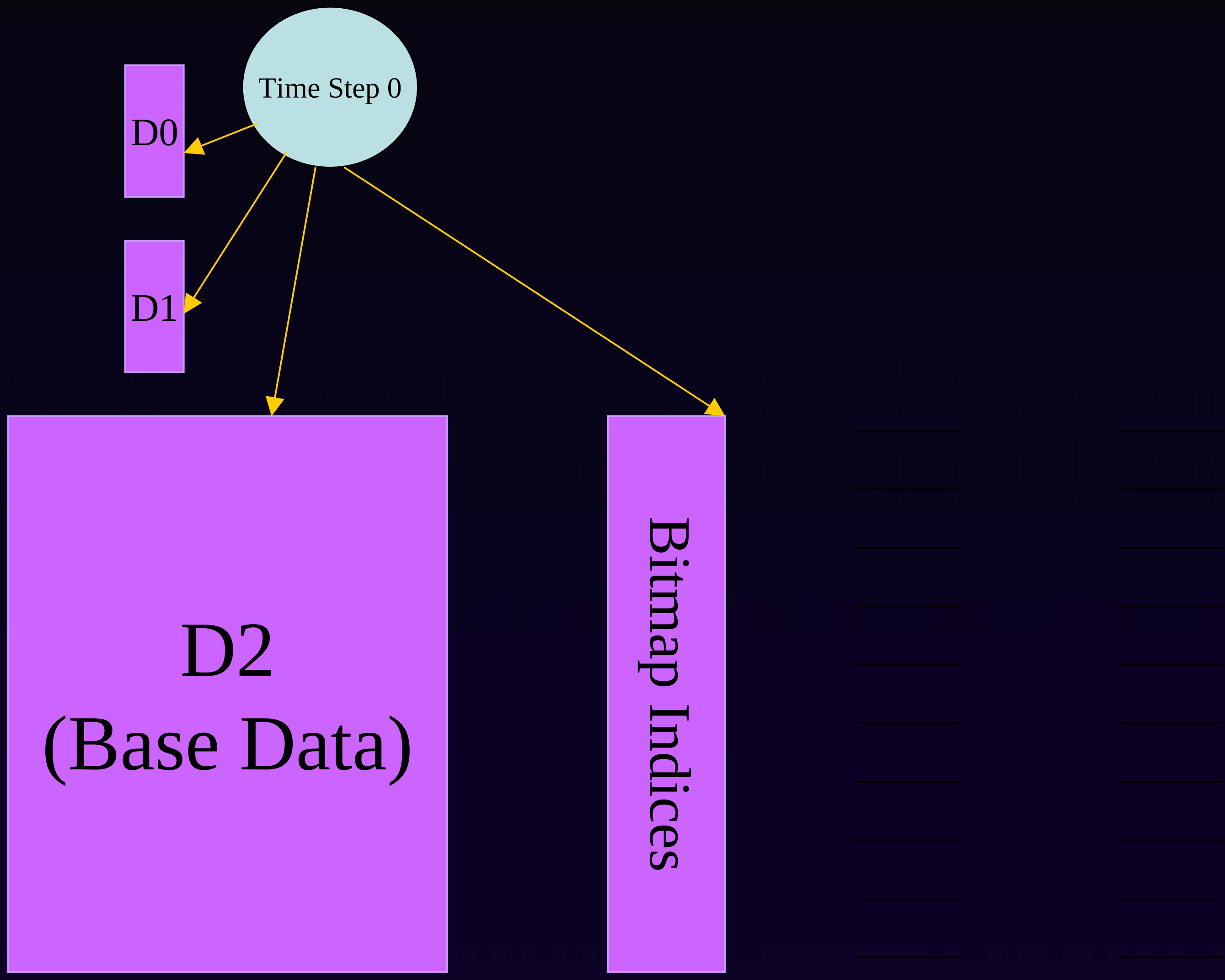
Some Simplifying Assumptions

- Block structured data
  - 0-3 Dimensional topology (arbitrary geometry)
  - Limited Datatypes: *float, double, int32, int64, byte*
  - Vectors and Tensors identified via metadata
- Two Level hierarchical organization
  - TimeStep
  - VariableName
  - Queries can be posed implicitly across time dimension
- Future
  - Arbitrary nesting
    - AMR "Level"
    - CalibrationSet
  - More Data Schemas
    - Unstructured
    - AMR
    - NetLogs

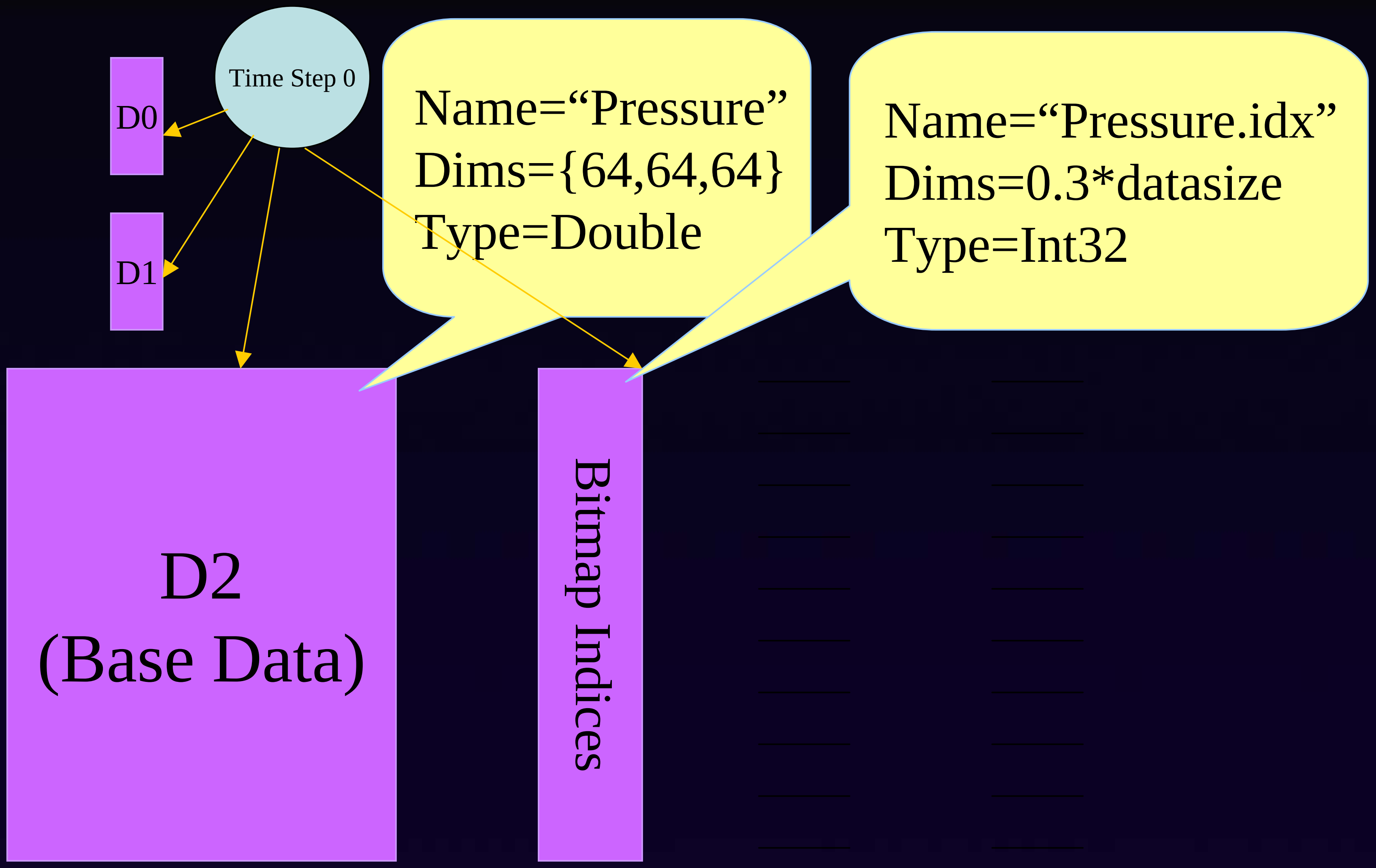
HDF5 Data Organization to Support FastQuery



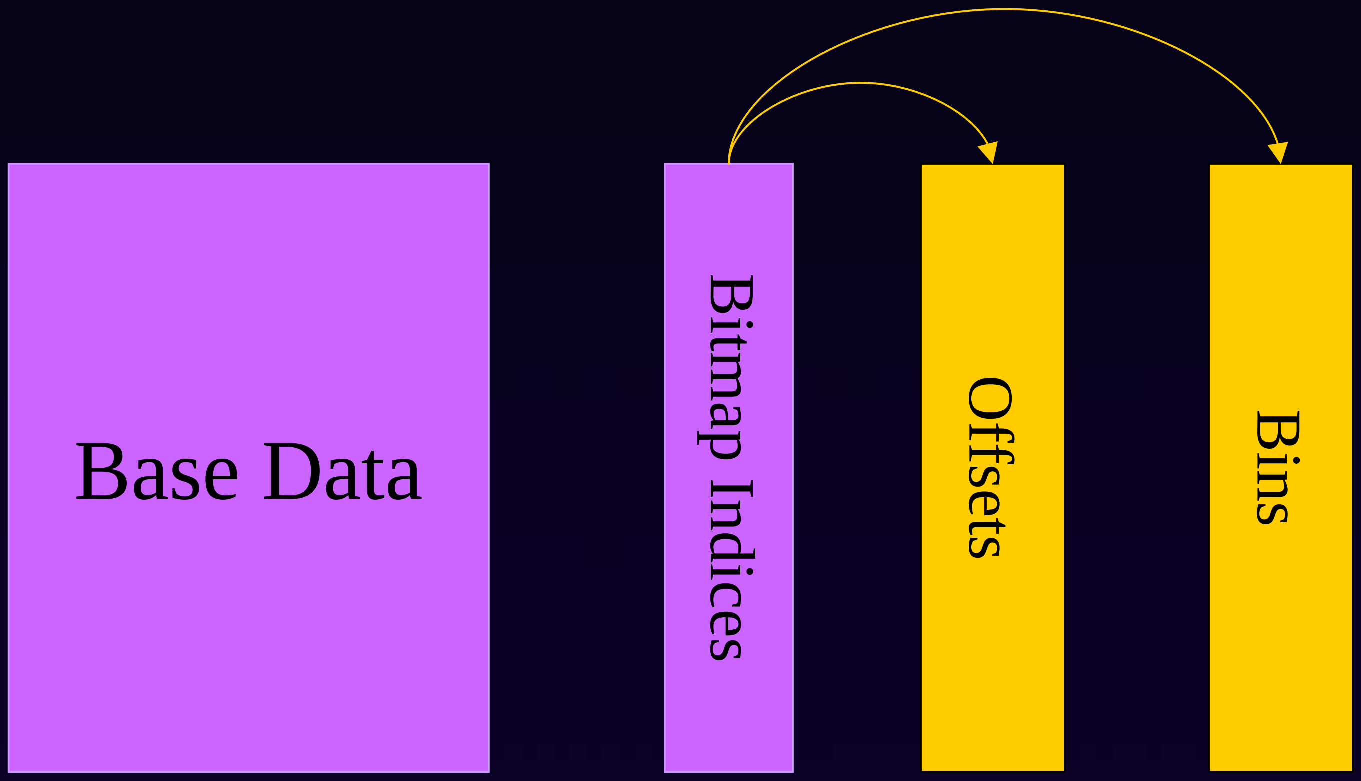
File Organization



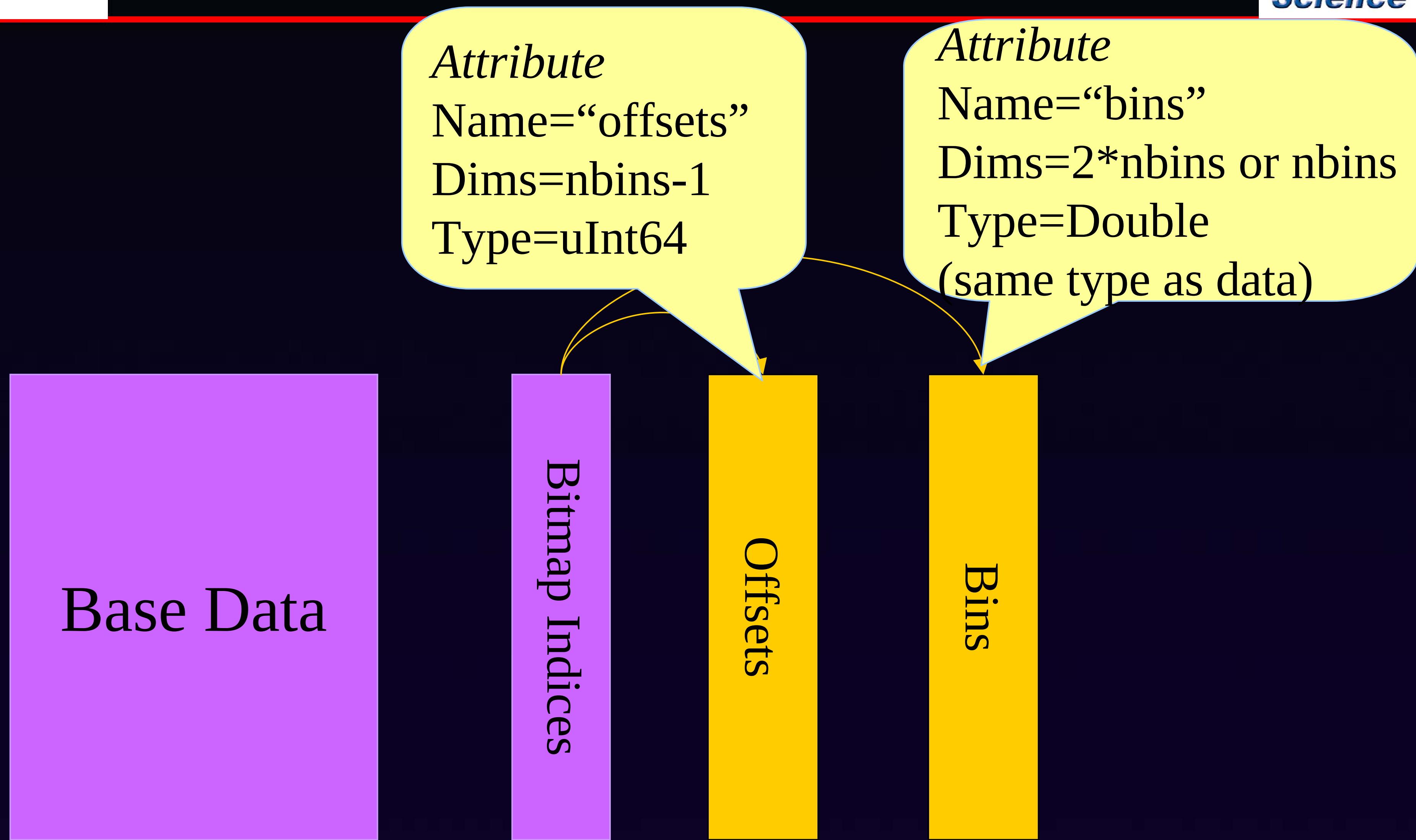
File Organization



File Organization

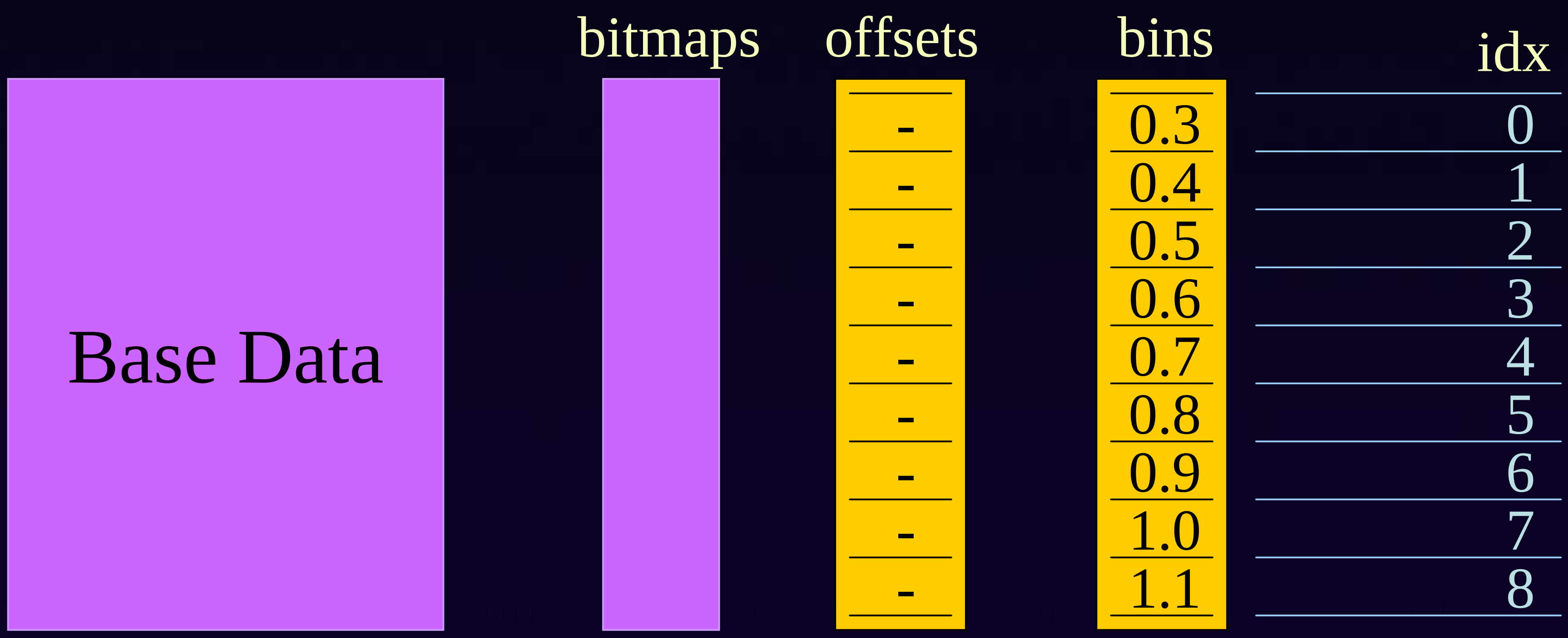


File Organization



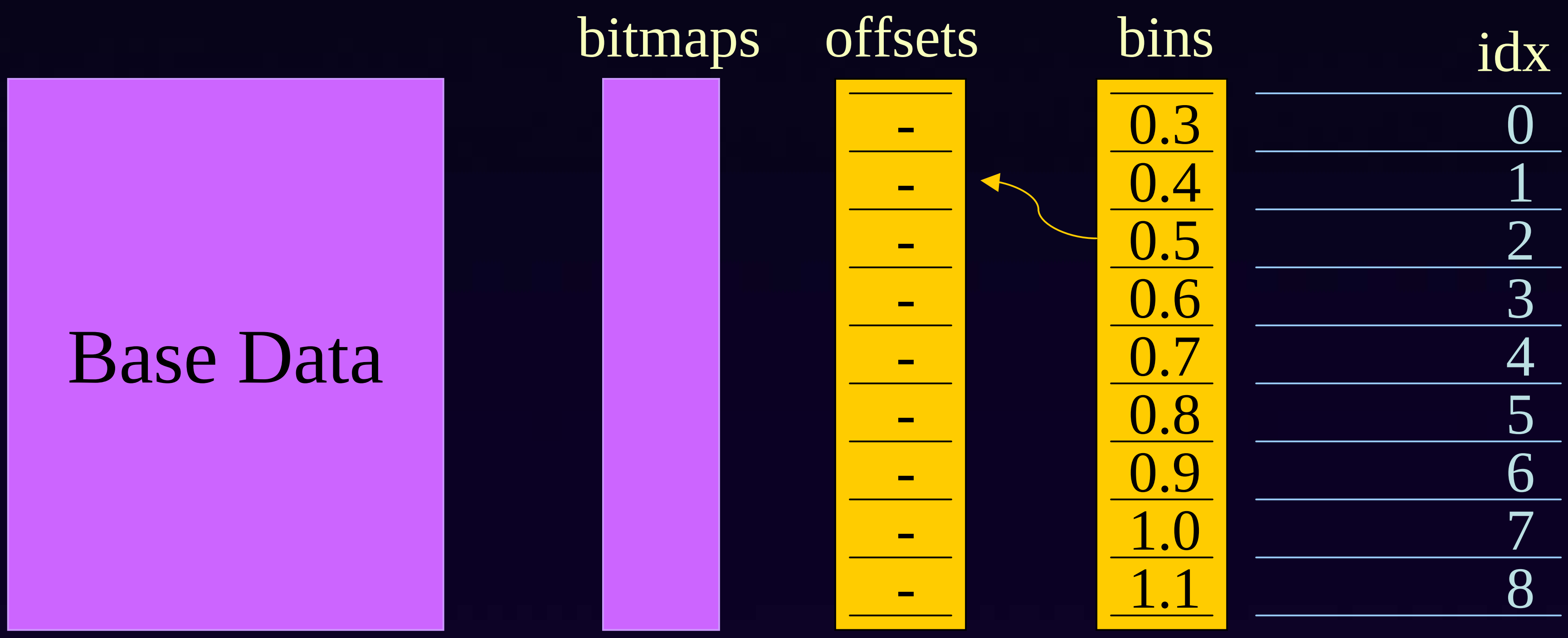
File Organization

Query: Pressure < 0.5



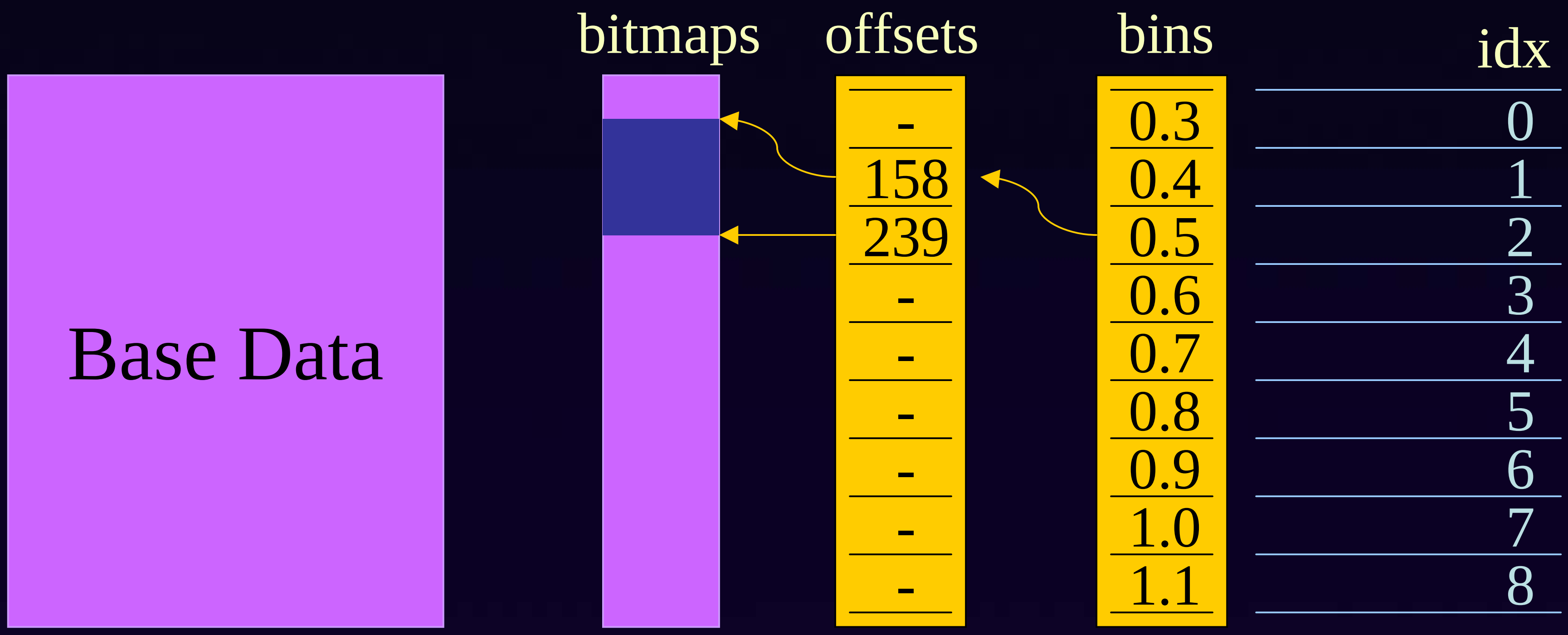
File Organization

Query: Pressure < 0.5



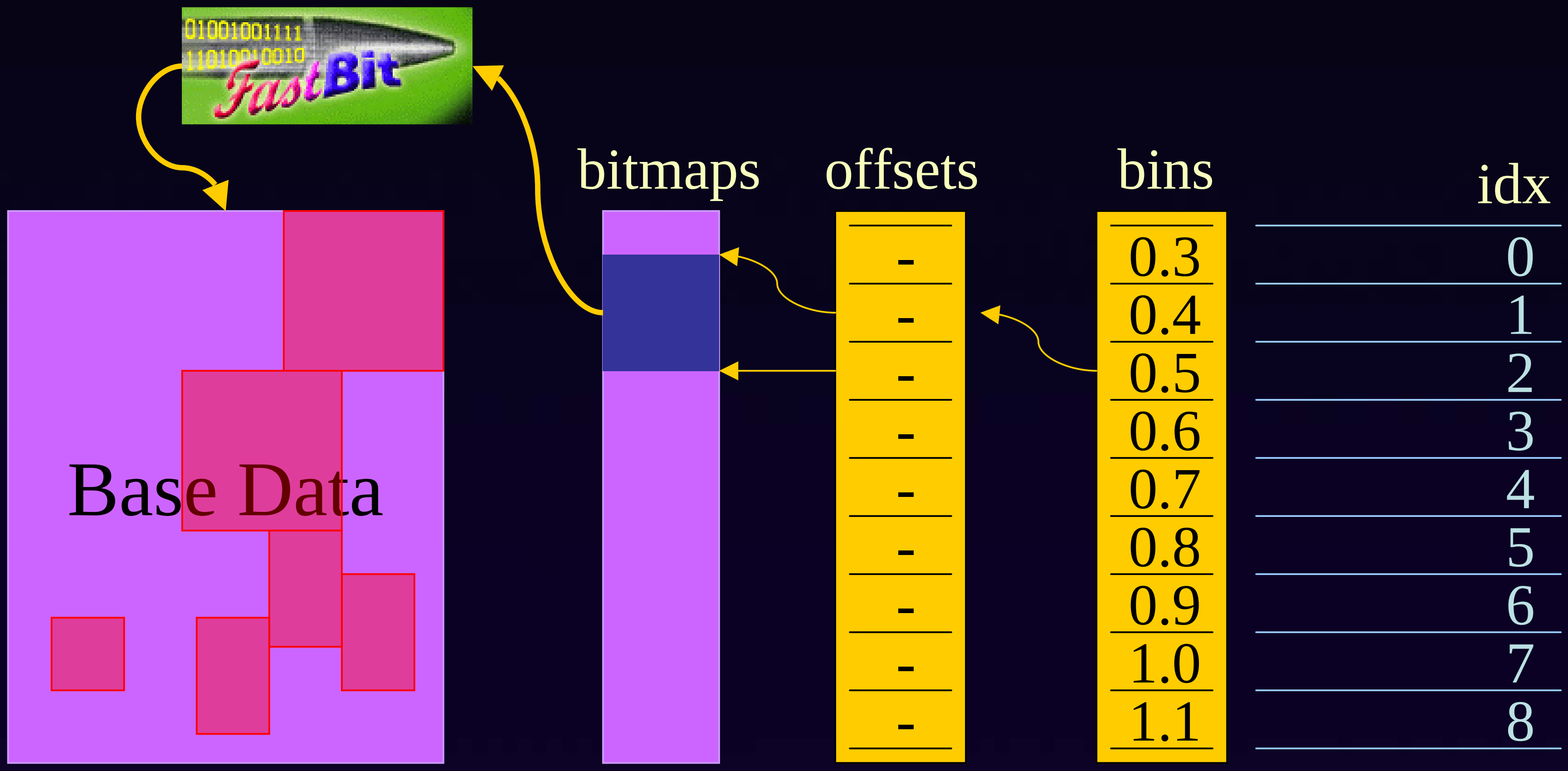
File Organization

Query: Pressure < 0.5



File Organization

Query: Pressure < 0.5



Final Notes

- Need for Higher level data organization
  - Demonstrated simple convention for index storage
  - Require higher level data organization to support more complex queries demanded by our scientific applications
  - Adoption of higher-level schema is a sociological problem rather than a technical problem
- Top Down (the Grand Unified Data Model)
  - DMF: Describe everything in the known universe
- Bottom up (community building)
  - Research Group: *Store data fro Cactus*
  - Scientific Community: *eg. HDF-EOS, NetCDF, FITS*
  - *?*

Final Notes

- Need for Higher level data organization
  - Demonstrated simple convention for index storage
  - Require higher level data organization to support more complex queries demanded by our scientific applications
  - Adoption of higher-level schema is a sociological problem rather than a technical problem
- Top Down (the Grand Unified Data Model)
  - DMF: Describe everything in the known universe
- Bottom up (community building)
  - Research Group: *Store data fro Cactus*
  - Scientific Community: *eg. HDF-EOS, NetCDF, FITS*
  - World Domination

Questions?

Performance of Event Catalog

- The Event Catalog uses compressed bitmap indices
  - The most commonly used index is B-tree
  - The most efficient one is often the projection index
- The following table reports the size and the average query processing time
  - 1-attribute, 2-attribute, and 5-attribute refer to the number of attributes in a query
- Compressed bitmap indices are about half the size of B-trees, and are 10 times faster
- Compressed bitmap indices are larger than projection indices, but are 3 times faster

2.2 Million Events 12 common attributes		B-tree	Projection index	Bitmap index
Size (MB)		408	113	186
Query processing (seconds)	1-attribute	0.95	0.51	0.02
	2-attribute	2.15	0.56	0.04
	5-attribute	2.23	0.67	0.17