An Exploration of SciDB in the Context of Emerging Technologies for Data Stores in Particle Physics and Cosmology

David Malon*, Peter van Gemmeren, Jack Weinstein

Argonne National Laboratory

•malon@anl.gov

ACAT 2011

5-9 September 2011

U.S. DEPARTMENT OF ENERGY

Disclaimers and acknowledgments

- Initial explorations of preliminary, “pre-alpha” releases of SciDB were conducted at Argonne in 2010, but …
- With a “real” release of SciDB available for the first time in June 2011, most of those early tests were redone and the bulk of the work described herein was undertaken by an Argonne summer student, Jack Weinstein, now at University of Illinois at Urbana-Champaign
  - Supervised by Peter van Gemmeren
  - Jack should be giving this talk … but classes have begun
- Thanks too to the SciDB team for two or three of their slides included in this presentation
- Misunderstandings and misrepresentations are entirely mine

Emerging database technologies and HEP

- HEP community uses databases extensively for a very wide range of purposes
- The community has developed its own technologies to address many database-related problems that the database research community has also been thinking about—usually for good reasons
  - Temporal databases? COOL
  - Distributed database content caching? FroNTier
  - …
- To many computational science applications in HEP, though, databases remain a less-than-good match for a variety of reasons
- Even for structures as simple as event-level metadata records, databases tend to be deficient
  - Scalable support for variable-length structures (e.g., information about varying numbers of jets, leptons, …, in each event)
  - Combinatorial operators
  - Decoding time-varying information (e.g., trigger decisions: same triggers aren't used in every run) requires work

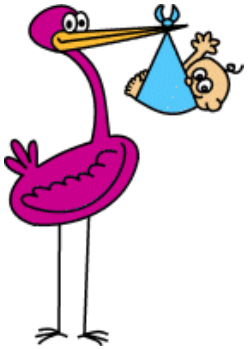
Even as the HEP database "software stack" for LHC-era experiments has gradually emerged, what else has been going on?

- Database technologies are evolving
- Column-wise databases are now not at all unusual
- Temporal database technologies have improved
- Distributed and replicated database approaches are everywhere
- HEP data scales are no longer rare, and others are addressing such data volumes
  - inside and outside of databases
- People are thinking about database access in cloud computing

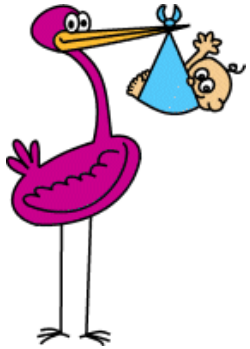
Are there emerging database technologies that the HEP community should be tracking?

SciDB: some history

- A couple of years ago, some of the biggest names in database academe asked themselves why big sciences aren't using databases for their primary data
- Aware of history; some of them were integrally involved
  - Pick your favorite limited success (or failure) story
- Of the many reasons for lack of success, things they heard repeatedly were
  - Need for arrays—hard to emulate in RDBMS—and even nested arrays
  - Lack of features: provenance, version control, internal support for uncertainty (error bars on data), …
  - Operations mismatch with SQL
- Seek to address the "need for a scalable open-source data management system with integrated analytics to support scientific research on massive data sets"



SciDB is born



Preserving ordering helps complex computations

Relational Database

I	J	value
0	0	32.5
1	0	90.9
2	0	42.1
3	0	96.7
0	1	46.3
1	1	35.4
2	1	35.7
3	1	41.3
0	2	81.7
1	2	35.9
2	2	35.3
3	2	89.9
0	3	53.6
1	3	86.3
2	3	45.9
3	3	27.6

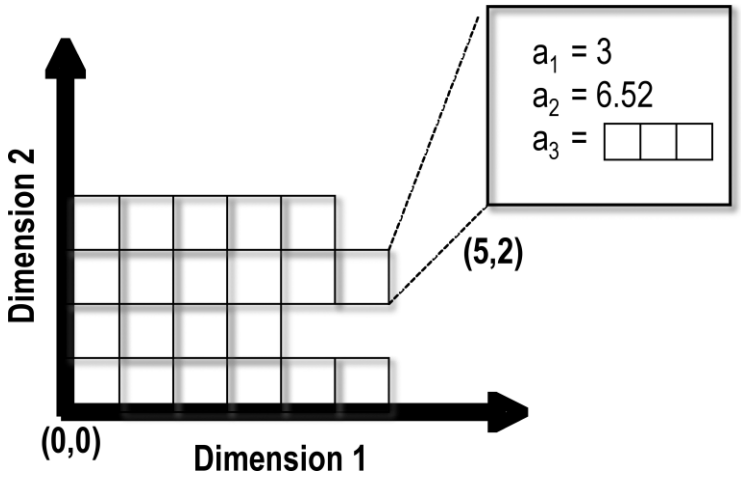
Paradigm4

32.5	46.3	81.7	53.6
90.9	35.4	35.9	86.3
42.1	35.7	35.3	45.9
96.7	41.3	89.9	27.6

- Speeds up data access in a distributed database
- Facilitates drill-down & clustering by like groups
- Math functions like linear algebra run directly on native storage format

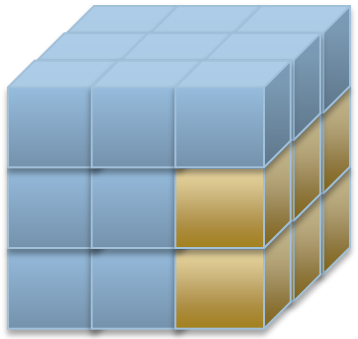
Array Data Model

- Nested multi-dimensional arrays
  - Cells can be tuples or other arrays
- Time is an automatically supported extra dimension
- Extensible type system
- Ragged arrays allow each row/column to have a different dimensionality
- Support for multiple flavors of ‘null’
  - Array cells can be ‘EMPTY’
  - User-definable, context-sensitive treatment



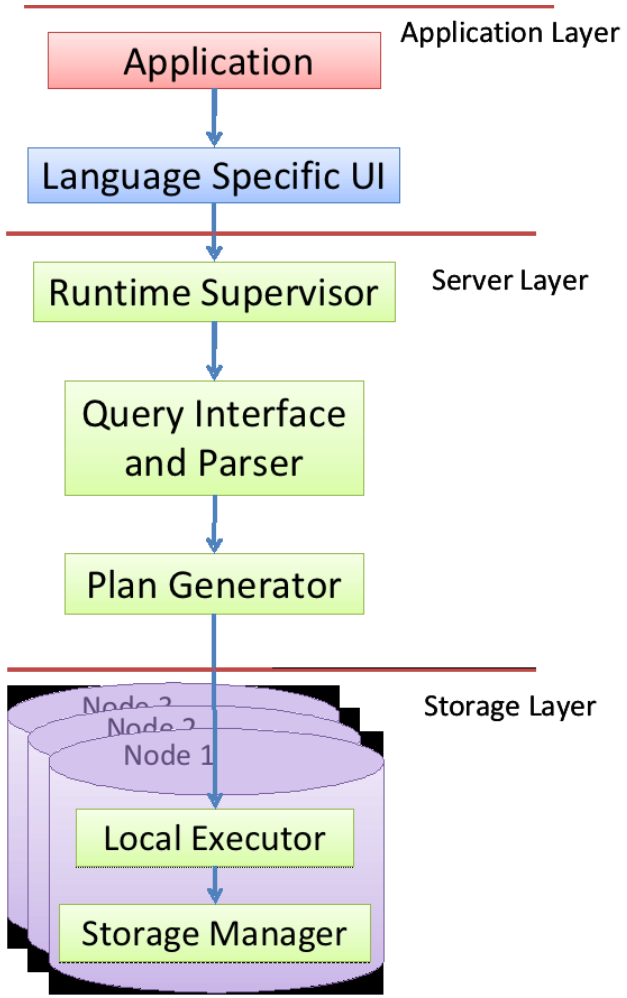
Data Storage

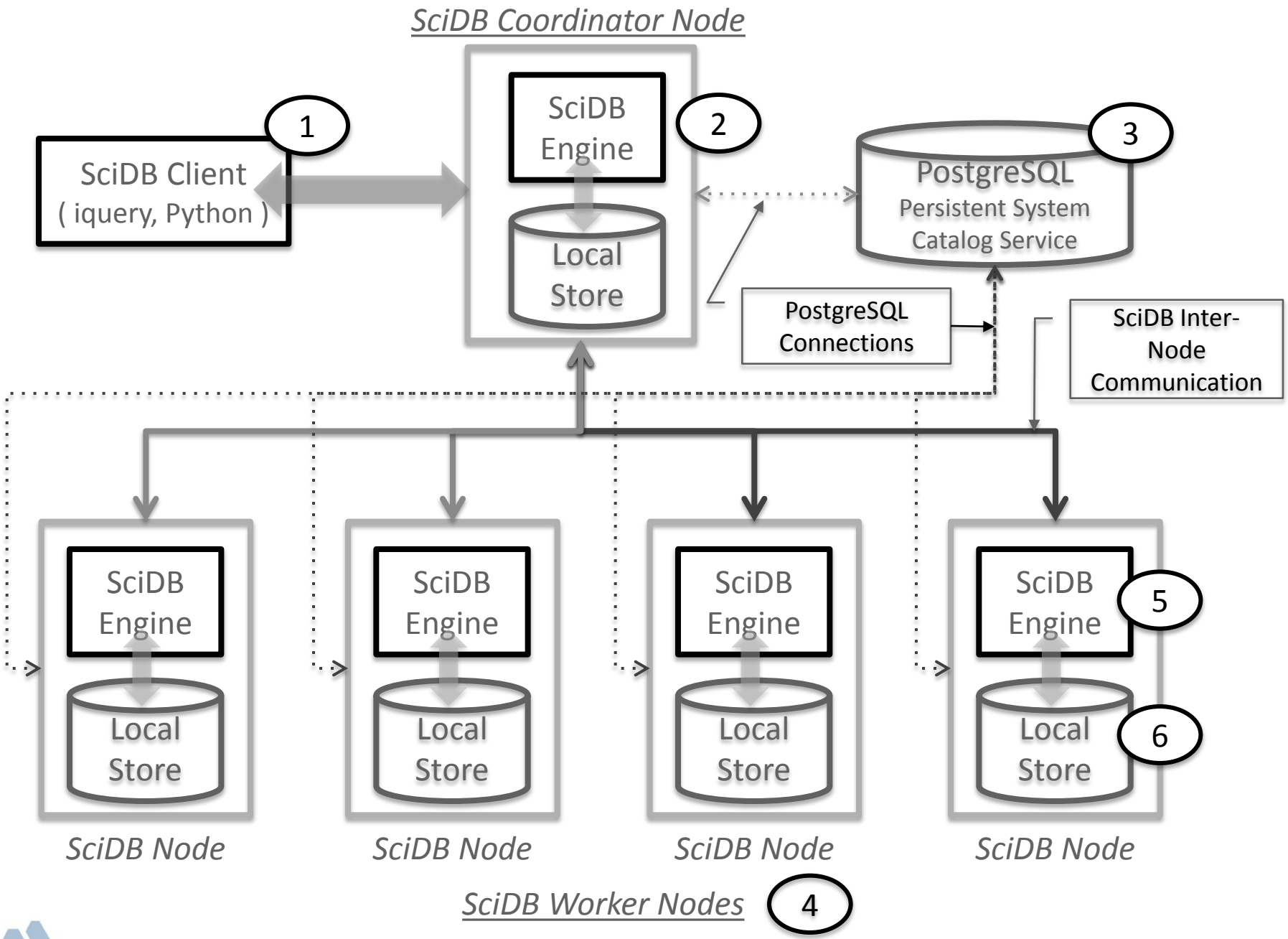
- Arrays are “chunked” (in multiple dimensions) in storage
- Chunks are partitioned across a collection of nodes
- Each node has processor and storage (shared nothing)
- Chunks have ‘overlap’ to support neighborhood operations



Architecture

- Shared nothing cluster
  - 10's–1000's of nodes
  - Commodity hardware
- Queries refer to arrays as if not distributed
- Query planner optimizes queries for efficient data access & processing
- Query plan runs on a node's local executor/storage manager
- Runtime supervisor coordinates execution




SciDB Coordinator Node
SciDB Client
(iquery, Python)
1
SciDB
Engine
2
Local
Store
3
PostgreSQL
Persistent System
Catalog Service
PostgreSQL
Connections
SciDB Inter-
Node
Communication
SciDB
Engine
Local
Store
SciDB Node
SciDB
Engine
Local
Store
SciDB Node
SciDB
Engine
Local
Store
SciDB Node
SciDB
Engine
5
Local
Store
6
SciDB Node
SciDB Worker Nodes
4

Query Language & Operators

- Array Query Language -- AQL
  - Declarative SQL-like language extended for working with array data
- Array Functional Language -- AFL
  - Functional language for internal implementation of the operators
- Array Specific Operators
  - Aggregate, Apply, Filter, Join, Regrid, Subsample, et al
- Linear Algebra, Statistical, and other applied math operators
  - Matrix operations, clustering, covariance, linear and logistic regression, et al
- Extensible with Postgres-style user-defined functions
  - e.g., to ‘cook’ images, perform customized search, FFT, feature-detect
- Interfaces to other open source packages
  - MatLab, R, SAS

Array Query Language (AQL)

```
CREATE ARRAY Test_Array
    < A: integer NULLS,
      B: double,
      C: USER_DEFINED_TYPE >
    [I=0:99999,1000, 10, J=0:99999,1000, 10 ]
    PARTITION OVER ( Node1, Node2, Node3 )
    USING block_cyclic();
```

attribute names: A, B, C

index names: I, J

chunk size: 1000

overlap: 10

Array Query Language (AQL)

```
SELECT Geo-Mean ( T.B )
FROM Test_Array T
WHERE
   T.I BETWEEN :C1 AND :C2
AND T.J BETWEEN :C3 AND :C4
AND T.A = 10
GROUP BY T.I;
```

User-defined aggregate on an attribute B in array T

Subsample

Filter

Group-by

So far as SELECT / FROM / WHERE / GROUP BY queries are concerned, there is little logical difference between AQL and SQL

But why should a proton collider experiment care (or not) about SciDB?

- Array-oriented operations with support for overlapping chunks—the focus of initial SciDB development—is *not* what is most of interest to us
  - Though in certain other domains (e.g., in astrophysics) this could be key
- Same parallel execution architecture, though, and explicit design to support multi-petabyte data stores, may help us
  - As may its explicit support for sparse data
- Design with a view of transactions that is aware of the write-once nature of most of our data is interesting
- SciDB plans to support *in situ* data, in addition to SciDB-native storage formats, are very interesting
  - Bulk data could be in a format known and proven within the community
  - Initially, NetCDF/HDF5 proposed, but no reason that, say, a ROOT storage backend could not be provided
- Support for user-defined functions and types could be extremely useful
- Provenance support (though initially primitive) is a plus
- Support for uncertainty in data is interesting in principle
  - A bit of skepticism, though, about how much this helps us in practice, may be in order

So?

- Shared-nothing architectures on commodity clusters?
  - Make sense, but there are lots of products that do this
- Array extensions to traditional SQL?
  - Worthwhile idea for some data, but do they help in experimental particle physics when array indices might be {run #, event #}, or {run #, event#, jet #}?

- Focus of student project was not upon performance, but upon evaluation of SciDB's capability of supporting HEP data and data types, and natural operations thereon

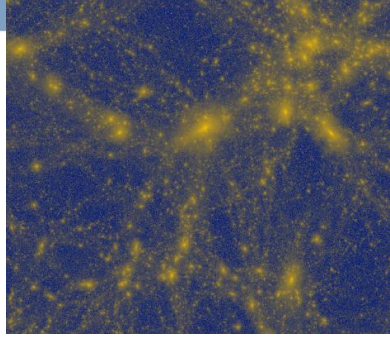
Scope of the project

- Install and configure first real release of SciDB, R11.06
- Provide machinery to upload data from multiple source types, including ROOT files
- Exercise the user interface and operators
- Explore tradeoffs in data organization
- Evaluate sparse versus dense data representations of identical data
- Provide user-defined types and functions in support of scientific data types

First, the data: event-level metadata from proton collisions at LHC

- ATLAS records key quantities describing properties of proton collision events as event-level metadata
- Metadata records contain information useful in event selection
  - Including information about *variable* numbers of jets, photons, muons, electrons, …
  - And triggers passed by the events
  - Along with sufficient navigational information to allow one to navigate to file-resident raw and derived event data for events that satisfy selection criteria
- Chosen for early tests because data structure is relatively simple, and event-level metadata records are already hosted both in files and in relational databases
  - Variable-length array data members are not a natural fit to relational databases, though they can be supported
  - And queries may be complex and combinatorial
- Data from 2011 ATLAS runs provided a basis for many of our tests

More data: cosmological simulations



- Data associated with cosmological simulations that can challenge the capabilities of the largest leadership computing facilities in fact often have very simple structure
  - N-body problem(s)
  - 8 floating point variables are stored for each particle (position, velocity, mass, id):
    - x[Mpc], v_x[km/s], y[Mpc], v_y[km/s], z[Mpc], v_z[km/s], particle mass[M_sol], particle tag
- Important derived data are not much more complex
- (Dark matter) halo catalogs
  - Halos are stored as halo_mass[10^10 M_sol], x[Mpc], y[Mpc], z[Mpc], vx[km/s], vy[km/s], vz[km/s
- Published reference datasets used in comparisons of cosmological simulations were uploaded in our tests
  - 256**3 particles
  - Natural candidates for exploiting three-vector constructs for position and velocity

Uploads

- New to SciDB: a Python “connector,” e.g.,

```
db = scidb.connect("localhost", 1239) # connect to the SciDB coordinator
db.executeQuery("drop array simplearray", 'aql') # execute an AQL query
```

- Python script developed to upload data from ROOT trees to SciDB
  - root2scidb.py
- Also a script to upload (FORTRAN direct I/O) cosmology data
- Bulk loads and parallel loading of chunks possible in principle

User-defined types and functions

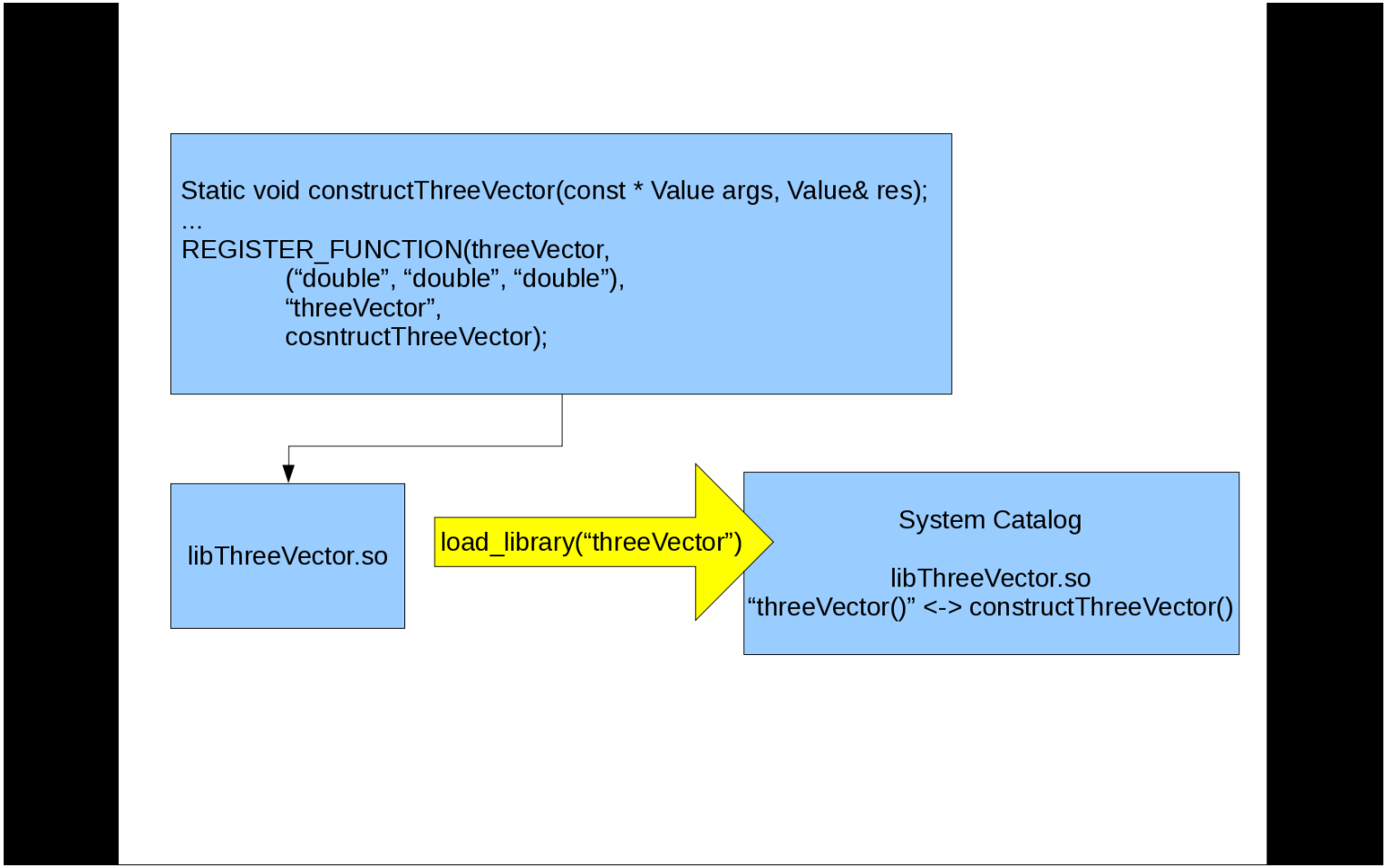
- Several user-defined types and functions were defined as part of this exercise
- Three-vector type
- Rotation functions
- Lorentz vector type
- Lorentz transformations
- … and a variety of related operators

- Examples:
  - apply(AttributeList, Jet, threeVector(JetPt1, JetEta1, JetPhi1))
  - sort(AttributeList, angleBetween(Jet1, threeVector(1,1,1)))

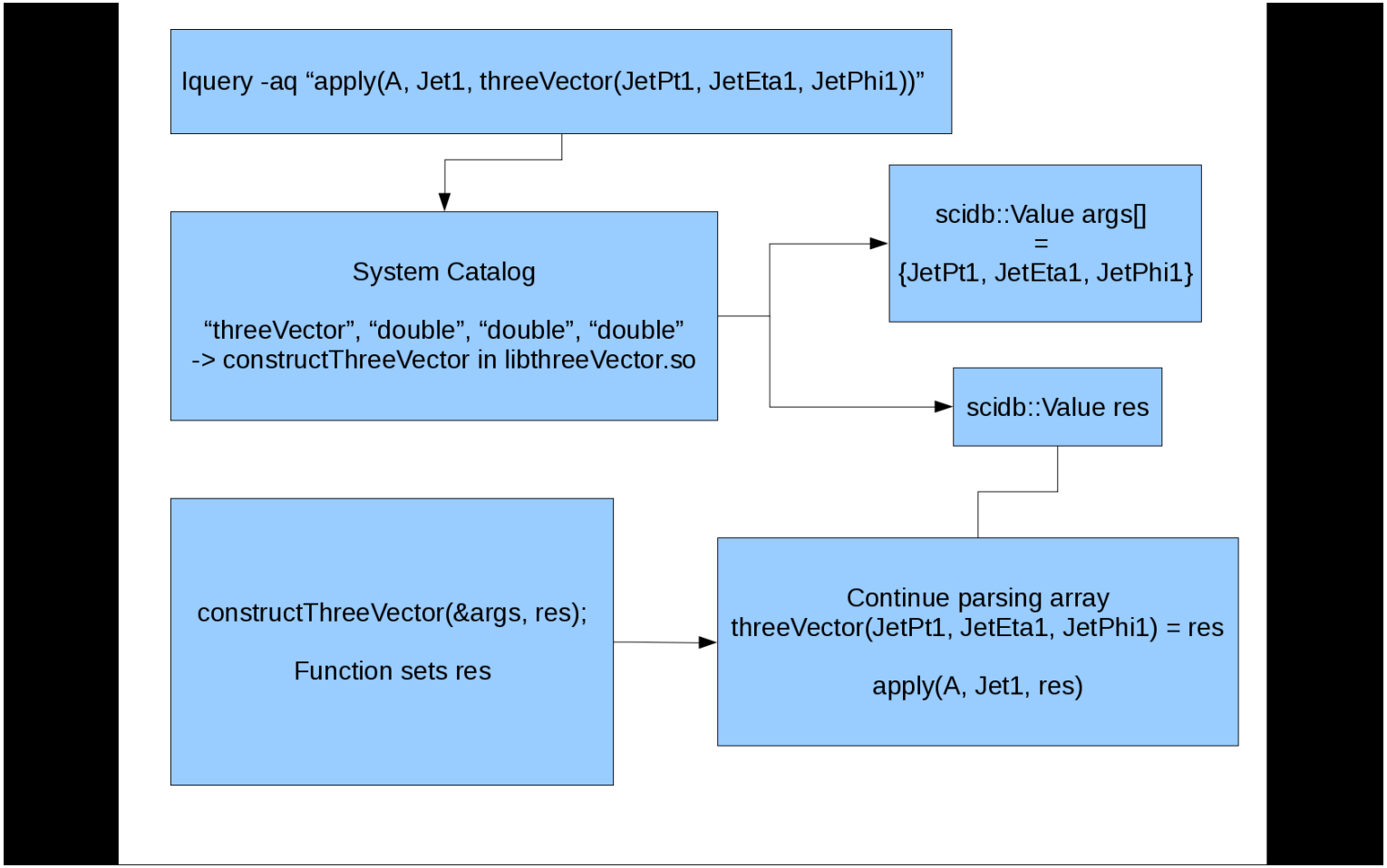
Under the covers…

- As usual when introducing a new type into a system, the tedious work is not defining the data members (a struct of three doubles, say)
- Rather, it is in
  - Defining all the supporting operators, e.g., for construction and assignment
  - Registering the type so that it is recognized and its appropriate associated operators invoked by the system into which it is introduced
  - And so on
- The following slides illustrate how this happens in SciDB

UDT supporting machinery: declaration and construction



UDT supporting machinery



First impressions

- Initial development aimed principally at supporting array data models, both in storage and in operations
  - Array data structures, partitioning (chunking), overlap specification, and basic operations thereon, are well underway
  - Most testable features are currently of this flavor
- For many sciences, this focus is entirely appropriate
- Designed to be well suited to image data
  - A major target client of SciDB is LSST
- Other development has necessarily received less emphasis to date
  - Including some basic relational database features, and planned non-array-oriented functionality

Status and prospects

- SciDB has advanced considerably in functionality and robustness since its earliest incarnations
- Most issues we encountered were minor, e.g.,
  - Configuration still exhibits certain inflexibilities, and SciDB is not difficult to crash
  - And we stop SciDB by broadcasting kill -9?
  - Printout and other user interface features still primitive
- Composition of functions works well, e.g.,

  iquery -p 1341 -q "project(filter(subsample(AttributeList, 0, 20), NJet>0), 'NJet')"
- Support for user-defined types and operators is functional, with some minor limitations
  - Limit of three arguments to type constructors
  - Loading user-defined types requires conversion to string but works well enough
- This is, of course, only a status snapshot, and continual improvement is expected
- HEP applications can learn and profit from architectures such as that of SciDB

Further work?

- See also work by Nikolay Malitsky at Brookhaven National Laboratory, who is investigating use of SciDB for EPICS (Experimental Physics and Industrial Control System) data
- And track SciDB and its applications at www.scidb.org