

Overview of SciDB: Large Scale Array Storage, Processing and Analysis

Presenter: Kevin Chang

15-799

10/30/2013

Executive Summary

- **Problem:**

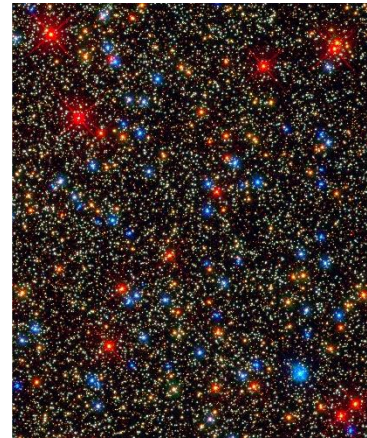
- Modern databases are primarily built for **business** applications
- **Scientific data** have different characteristics and are becoming more data-intensive

- **Solution:** SciDB

- A new open-source DBMS to support VERY large (PB) **ARRAY** data

Background and Motivation

- Scientific data differ from business data in 3 ways
- **1. Science makes use of *sensor arrays***
 - Rectangular array of individual sensors
 - Example: Hubble Space Telescope's camera



- A block of collocated pixels
- Implicit ordering

Background and Motivation

- **2. Require sophisticated data processing methods**
 - Sensor data needs filtering
 - Clean data are fed into complex analytic processing

- **3. Extremely LARGE data**
 - Produced at large scale and reused frequently
 - Example: LHC -> 15PB data annually

SciDB Features

- Goal: provide scalability to store, process and analyze data
- Array data model:
 - Implicit ordering
 - Organized as collections of n-dimensional arrays
 - Each cell contains a **tuple of values** with attribute names

(A::INTEGER, B::FLOAT) -> (2, 0.7)

Array Definition

CREATE ARRAY EXAMPLE

(A::INTEGER, B::FLOAT) [I=0:2, J=0:2];

I = 0 : 2

(2, 0.7)	(1, 0.5)	(3, 0.1)
(6, 0.3)	(7, 0.1)	(0, 0.1)
(1, 0.4)	(4, 0.6)	(8, 0.1)

I = 0 : 2

J = 0 : 2

(2, 0.7)	(1, 0.5)	(3, 0.1)
	(7, 0.1)	(0, 0.1)
(1, 0.4)	(4, 0.6)	

Empty cells

- SciDB will support extensible type system (PDF)
- Nested model (an array within an array)

Data Manipulation

- Declarative query language

Slice (Example, $l=2$);

(2, 0.7)	(1, 0.5)	(3, 0.1)
(6, 0.3)	(7, 0.1)	(0, 0.1)
(1, 0.4)	(4, 0.6)	(8, 0.1)



(3, 0.1)
(0, 0.1)
(8, 0.1)



**Subsample (Example,
I BETWEEN 0 AND 1 AND
J BETWEEN 0 AND 1);**

(2, 0.7)	(1, 0.5)
(6, 0.3)	(7, 0.1)

Data Manipulation

**Sjoin (Slice (Example, l=2),
Subsample (Example,
I BETWEEN 0 AND 1 AND
J BETWEEN 0 AND 1)
);**

(2, 0.7)	(1, 0.5)
(6, 0.3)	(7, 0.1)

(3, 0.1)
(0, 0.1)
(8, 0.1)



(2, 0.7)	(1, 0.5)	(3, 0.1)
(6, 0.3)	(7, 0.1)	(0, 0.1)
		(8, 0.1)

Data Manipulation

- Manipulate data content

Filter (Example, $A > 2$);

- Operations are *composable*

Apply (Slice (Example, $l = 2$), $A * B$);

Structural operator



Content manipulator



Extensibility

- User-defined data type (UDT)
- User-defined function (UDF)
 - Highly specific algorithm
 - C/C++
 - Example: **Gaussian smoothing**

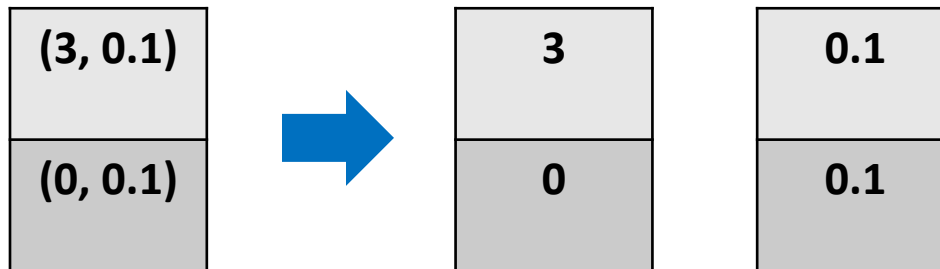
**Smooth (Subsample (Example,
I BETWEEN 0 AND 1 AND
J BETWEEN 0 AND 1));**

Architecture

- Shared nothing over a network of computers
- One centralized system catalog database
 - Stores info about data distribution, UDF, etc
- Storage manager:
 - Distributed and **no-overwrite** storage manager
 - Data can only be appended
 - ACID: **Atomicity** and **durability**
 - Does not talk about its implementation

Data storage

- Mapping of logical array data into physical storage
- **1. Vertical partitioning like column store:**
 - Users tend to focus on a single attribute
 - Each cell only stores a single value

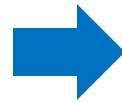


Data Storage

- 2. Chunking:

- Operate on a small area
- Split into equal-sized (and overlapping) chunks

(2, 0.7)	(1, 0.5)	(3, 0.1)
(6, 0.3)	(7, 0.1)	(0, 0.1)
(1, 0.4)	(4, 0.6)	(8, 0.1)

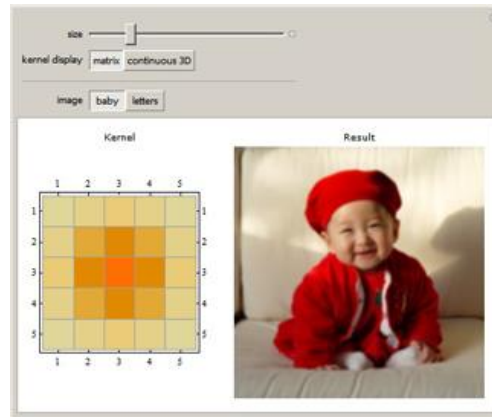


(2, 0.7)	(1, 0.5)	(1, 0.5)	(3, 0.1)
(6, 0.3)	(7, 0.1)	(7, 0.1)	(0, 0.1)
(6, 0.3)	(7, 0.1)	(7, 0.1)	(0, 0.1)
(1, 0.4)	(4, 0.6)	(4, 0.6)	(8, 0.1)

Data Storage

- Overlapping chunks:

- Some methods require computation on neighboring cells
- Example: Gaussian smoothing



- Advantage: enables parallel computation
- Storage overhead Apply (Slice (Example, I = 2), $A * B$);
- Pipelined operators

Other Features

- **Data distribution**: Some chunks are collocated on the same node for *locality*
- **Compression**: Reduce bandwidth consumption
 - Sparse matrix compression?

Query Processing

- Complicate parsing due to user-defined operators
- Optimization: Tries to parallelize queries
- Scatter/Gather operations to dynamically redistribute data across nodes

Conclusion

- A new **ARRAY** DBMS
 - Data management
 - Math operations
- Open source
- SciDB outperforms Postgres by 2 orders of magnitude
 - Astronomy-style workloads

THE END