Group 19:

Smitha Malur Muralidhar
Purva Kolhatkar
Manasi Pradhan
Matthew Tschiggfrie
Outline

1. Why SciDB?
2. SciDB Architecture
3. SciDB-Py
4. SciDB-R
5. Popular Applications
6. Advantages and Disadvantages
What does scientific data look like?

- Extensive use of sensor arrays
- Scientific analysis involves sophisticated data processing.
- Data is large and is reused.
Why sciDB?
Why sciDB?

- Inadequacy of current commercial DBMS
- Custom database for every project.
- Natural relational table model doesn’t suit scientific data.
- Science community was reluctant to learn new programming language.
Who developed SciDB?

2008: Multi-institution project.

What is sciDB?

- Open source
- Distributed array database
- Horizontally scalable
- In database math
- ACID
- Integrated with R and python
SciDB Architecture
Arrays

2D Array Arrangement

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>row[0]</td>
<td>10</td>
<td>20</td>
<td>45</td>
</tr>
<tr>
<td>row[1]</td>
<td>42</td>
<td>79</td>
<td>81</td>
</tr>
<tr>
<td>row[2]</td>
<td>89</td>
<td>9</td>
<td>36</td>
</tr>
</tbody>
</table>

Three-dimensional array with twenty four elements
Array Data Model: Terminology Used

- **Attributes**
  - Price: 450.61
  - Volume: 150
  - Symbol: “AAPL”
  - usec: 36013008713

- **Table in Relational DBMS**

- **Dimensions**
  - $a_1 = 3$
  - $a_2 = 6.52$
  - $a_3 = \text{missing}$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dash</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Bailey,</td>
<td>9.84</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Greene,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9.85</td>
<td></td>
</tr>
<tr>
<td>(Bolt,</td>
<td></td>
<td></td>
<td></td>
<td>9.69</td>
</tr>
<tr>
<td>(Kosgei,</td>
<td></td>
<td>508.17</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Kipruto,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Kemboi,</td>
<td></td>
<td></td>
<td>485.81</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Wanjiru,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Abera,</td>
<td>7811</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Baldini,</td>
<td>7855</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Wanjiru,</td>
<td>7596</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Thugwane,</td>
<td>7956</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Re-dimensioning arrays

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Symbol</th>
<th>Price</th>
<th>Volume</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>“AAPL”</td>
<td>450.61</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>“AAPL”</td>
<td>450.73</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>“AAPL”</td>
<td>450.84</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>“MSFT”</td>
<td>36.57</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>“MSFT”</td>
<td>36.20</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Symbol</th>
<th>Price</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>“AAPL”</td>
<td></td>
<td>450.61</td>
<td>150</td>
</tr>
<tr>
<td>“AAPL”</td>
<td></td>
<td>450.73</td>
<td>200</td>
</tr>
<tr>
<td>“AAPL”</td>
<td></td>
<td>450.84</td>
<td>10</td>
</tr>
<tr>
<td>“MSFT”</td>
<td></td>
<td>36.57</td>
<td>75</td>
</tr>
<tr>
<td>“MSFT”</td>
<td></td>
<td>36.20</td>
<td>100</td>
</tr>
</tbody>
</table>
Range selection in Relational database

<table>
<thead>
<tr>
<th>I</th>
<th>J</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>32.5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>90.9</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>42.1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>96.7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>46.3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>35.4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>35.7</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>41.3</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>81.7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>35.9</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>35.3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>89.9</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>53.6</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>86.3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>45.9</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>27.6</td>
</tr>
</tbody>
</table>

48 cells

Range selection in SciDB

Array Database

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>32.5</td>
<td>46.3</td>
<td>81.7</td>
<td>53.6</td>
</tr>
<tr>
<td>90.9</td>
<td>35.4</td>
<td>35.9</td>
<td>86.3</td>
</tr>
<tr>
<td>42.1</td>
<td>35.7</td>
<td>35.3</td>
<td>45.9</td>
</tr>
<tr>
<td>96.7</td>
<td>41.3</td>
<td>89.9</td>
<td>27.6</td>
</tr>
</tbody>
</table>

16 cells
SciDB
chunks
Multidimensional Array Clustering

- Chunks
- User defined co-ordinate system

```
CREATE ARRAY
STOCK_MARKET<PRICE: DOUBLE, VOLUME: DOUBLE>
[STOCK(string)
TIME(datetime)];
```
Architecture
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>false</td>
<td>Boolean value, true (1) or false (0)</td>
</tr>
<tr>
<td>char</td>
<td>\0</td>
<td>Single ASCII character</td>
</tr>
<tr>
<td>datetime</td>
<td>1970-01-01 00:00:00</td>
<td>Date and time</td>
</tr>
<tr>
<td>datetimetz</td>
<td>1970-01-01 00:00:00 -00:00</td>
<td>Date and time with timezone offset.</td>
</tr>
<tr>
<td>double</td>
<td>0</td>
<td>Double-precision floating point number</td>
</tr>
<tr>
<td>float</td>
<td>0</td>
<td>Single-precision floating-point number</td>
</tr>
<tr>
<td>int8</td>
<td>0</td>
<td>Signed 8-bit integer</td>
</tr>
<tr>
<td>int16</td>
<td>0</td>
<td>Signed 16-bit integer</td>
</tr>
<tr>
<td>int32</td>
<td>0</td>
<td>Signed 32-bit integer</td>
</tr>
<tr>
<td>int64</td>
<td>0</td>
<td>Signed 64-bit integer</td>
</tr>
<tr>
<td>string</td>
<td>&quot;&quot;</td>
<td>Variable length character string, default is the empty string</td>
</tr>
<tr>
<td>uint8</td>
<td>0</td>
<td>Unsigned 8-bit integer</td>
</tr>
<tr>
<td>uint16</td>
<td>0</td>
<td>Unsigned 16-bit integer</td>
</tr>
<tr>
<td>uint32</td>
<td>0</td>
<td>Unsigned 32-bit integer</td>
</tr>
<tr>
<td>uint64</td>
<td>0</td>
<td>Unsigned 64-bit integer</td>
</tr>
</tbody>
</table>
AQL and AFL

- Array Query Language
  - Data Definition Language: create and load arrays
  - Data Manipulation Language: select and operate on data stored in arrays

- Array Functional Language
  - Operators
    - Aggregate
    - Combine
    - Compute
    - Math
    - Rearrange
CREATE ARRAY Simple_Array <a1: double, a2: int64, a3: string> [I = 0 : *, 5, 0; J = 0 : 9, 5, 0];

Color index:
Attributes: a1, a2, a3
Dimensions: I, J
Dimension size: * is unbounded
Chunk size

SELECT a1 FROM Simple_Array;

SELECT I FROM Simple_Array;

INSERT INTO Array1
Select * from Array2
• CREATE ARRAY A <X: double,
    Y: double>
    [I = 0:99, 5, 0];

• CREATE ARRAY B <M: double,
    N: double>
    [I = 0:*, 5, 0  J = 0:99, 5, 0];

• Re-dimensioning array A:
  REDIMENSION_STORE(A, B);

• Aggregate operation: aggregate(A, count(X));
CREATE TABLE INPUT_A ( ROW INTEGER NOT NULL, COL INTEGER NOT NULL, VAL DOUBLE PRECISION, PRIMARY KEY ( ROW, COL ) );

CREATE TABLE INPUT_B ( ROW INTEGER NOT NULL, COL INTEGER NOT NULL, VAL DOUBLE PRECISION, PRIMARY KEY ( ROW, COL ) );

CREATE TABLE BASE ( ROW INTEGER NOT NULL, COL INTEGER NOT NULL, VAL DOUBLE PRECISION DEFAULT 0.0, PRIMARY KEY ( ROW, COL ) );

WITH MULTIPLY AS ( SELECT A.ROW, B.COL, SUM ( A.VAL * B.VAL ) AS VAL
FROM INPUT_A AS A
JOIN INPUT_B AS B ON A.COL = B.ROW
GROUP BY A.ROW, B.COL )
SELECT MULTIPLY.VAL + BASE.VAL
FROM MULTIPLY JOIN BASE ON MULTIPLY.ROW = BASE.ROW AND MULTIPLY.COL = BASE.COL;
input_A < val : double >[ row=0:4, col=0:5 ]
input_B < val : double >[ row=0:5, col=0:3 ]
base < val : double>[ row=0:4, col=0:3 ]
gemm ( input_A, input_B, base );

Corresponding query in SciDB
SciDB-py

Python library for SciDB

Easily store and grab arrays

Uses functions to load
SciDB-py Requirements

SciDB installation

Shim (network interface)

Python NumPy

shim: A Simple HTTP Service for SciDB

B. W. Lewis blewis@paradigm4.com
11/9/2015

- What’s new (for SciDB 15.12)
  - Support for the SciDB advanced I/O toolbox (aio_tools)
  - SciDB native authentication
  - Streaming and compression options no longer supported

- Overview

- Configuration
  - Ports and Network Interfaces
  - SciDB Port
  - Instance
  - Temporary I/O space
  - User
  - Max sessions
Upload Array to SciDB

- `from_array()`
- Uploads a numpy array
- Creates a SciDBArray object in python

```python
random() to create an array of uniformly distributed random floating-point values:
>>> # Create a 10x10 array of numbers between -1 and 2 (inclusive)
>>> # sampled from a uniform random distribution.
>>> A = sdb.random((10,10), lower=-1, upper=2)

randint() to create an array of uniformly distributed random integers:
>>> # Create a 10x10 array of uniform random integers between 0 and 10
>>> # (inclusive of 0, non-inclusive of 10)
>>> A = sdb.randint((10,10), lower=0, upper=10)

arrange() to create an array with evenly-spaced values given a step size:
>>> # Create a vector of ten integers, counting up from zero
>>> A = sdb.arange(10)

linspace() to create an array with evenly spaced values between supplied bounds:
>>> # Create a vector of 5 equally spaced numbers between 1 and 10,
>>> # including the endpoints:
>>> A = sdb.linspace(1, 10, 5)

identity() to create a sparse or dense identity matrix:
>>> # Create a 10x10 sparse, double-precision-valued identity matrix:
>>> A = sdb.identity(10, dtype='double', sparse=True)
```
Persistent Arrays

- New array functions take an argument called “persistent”.
- Persistent defaults to false.
  - True -> arrays stored in SciDB until removed
  - False -> arrays get removed after python session ended.
Accessing SciDB Array Objects

toarray()
todataframe()
tosparse()
Advantages of Using SciDB-py

- Python
- Aggregates
- No SQL queries
- Much like numpy
SciDB and R
Why R?

- Parallel computing in an easy way.
- Approach naturally fits analytics environment.
SciDB package for R

- Two main ways to interact with sciDB
- Use sciDB query language optionally returning results in data frames that can be iterated over.
- Use Array and dataframe like classes in R- statements backed by sciDB arrays
• Iquery executable → basic command line tool for communicating with sciDB
library (" scidb ")
library('threejs')
library('ggplot2')

source('/home/scidb/vm_functions.R')
#Will output "creating a generic function for 'image'... that is normal

scidbconnect ()

svded = scidb("KG_VAR_SVD")
# svded is an R representation of SciDB array KG_VAR_SVD

str(svded)
#outputs the structure of the R- representation of the array.
# Download just the 3 left vectors into R and make a matrix out of them:
svd_top = df2xyvm(iqdf(subset(svded, i<=2), n=Inf))

# Do kmeans clustering of these vectors in R now:
clustering = kmeans(svd_top, 5, nstart=50)

# Convert the kmeans cluster assignments to colors
color=gsub("[0-9]","",palette()[clustering$cluster+1])

# The relative distance between the dots is a measure of "genetic closeness"
print(qplot(x=svd_top[,1], y=svd_top[,2], color=I(color)))
# Vectors 1 and 3
qplot(x=svd_top[,1], y=svd_top[,3], color=I(color))
# Vectors 2 and 3
qplot(x=svd_top[,2], y=svd_top[,3], color=I(color))
Sample R scripts
Advantages SciDB-R

- Use SciDB as back-end database
- Use SciDB to offload large computations to cluster.
- Use SciDB to filter and join data before performing analytics.
- Use SciDB to share data among multiple users.
- Use SciDB to perform multi-dimensional windowing and aggregation.
Popular Applications

Early use cases - Resulted in birth and initial steps of SciDB

- Satellite Imagery
- Astronomy
- Genomics
Satellite Imagery - MODIS data

• Raw imagery of Earth data is a 3D array.
• Need to be fed into high level applications.
• Usually, the result is not satisfactory.
Astronomy - LSST data

- Telescope records images as 2D array.
- Lyra astronomy project needs a common repository for multiple telescopes.
- Need to be fed into high level applications.
Genomics

- Complete genome for a single human - 2D array
- Will be compared against human disease characteristics
- Biclustering in a large data set implemented in R vs implemented in SciDB-R showed significant differences.
Popular Applications

More refined uses - Resulted in growth of SciDB

- 1000 Genomes Browser
- LUX detector data
- Brazilian rainforests’ research
Theoretically, genotype data can be a 2D array.

Output of querying this data set is typically all columns for a row, or all rows for a column obtained by using slice and between operations.

Thus, array form of SciDB enables complex combinations of filter and cross_join queries.
LUX Detector by NERSC

- To gather evidence about the interaction between dark matter and normal matter.
- Represented as a 3D array, with 50 data attributes per cell.
- Complex queries involved like regrid, filter and cross_join.
- Using SciDB, entire analysis on 600,000,000 pulses took 4 hours.
Brazilian Rainforests’ Research by INPE

- An attempt to reproduce a controversial finding published by a different team.
- MODIS HDF-5 data set containing visible and infrared bands covering Brazil was used.
- Represented as 3D array - 7 TB data.
- SciDB took 4.6 hours to reproduce the finding.
Paradigm 4 customers:
Advantages of SciDB

• Keeps all the data
• Fast computation time
• Multiple instances
• No set data format
• Returns window query results in constant time
Advantages of SciDB over other systems

- RDBMS: Array system instead of tables.
- Fast data regridding
- In-situ linear algebra operations
- Science-appropriate operators in AQL
- Support for ‘never discard data’ policy of the scientific data users
- Can store uncertain nature of the scientific data
- Multiple types of “null” operator
Advantages of SciDB over other systems (contd):

- **File System:**
  - Metadata is not needed to be stored separately
  - Usual DBMS operations are used.
  - Exact layout of the file system is not needed to be known.

- **Hadoop:**
  - Has an efficient communication model
  - Not vulnerable to scalability issues
Disadvantages of SciDB

• Keeps all the data
• Small community
• Can’t organize arrays and metadata
• Not useful in small industries, small datasets and structured data
• Sparse dataset
Verdict for SciDB

To Use or Not To Use?
References

Thank you
HAVE ANY QUESTIONS

DO YOU?