SciDB: an open-source, array-oriented database management system

Philippe Cudré-Mauroux
Massachusetts Institute of Technology / University of Fribourg, Switzerland

& the SciDB Team

October 29, 2010
CIS, Itaipava, Brazil
I. Context
- Exascale Data Deluge
- XLDB Workshops
- The SciDB Consortium

II. System Overview
- Data Model
- Storage Model
- Operators
- Optimal Chunking
- Array Versioning
- SS-DB: a Science DBMS benchmark
Exascale Data Deluge

- Science
  - Biology
  - Astronomy
  - Remote Sensing
- Web companies
  - Ebay
  - Yahoo
- Financial services, retail companies, governments, etc.

➡ New data formats
➡ Peta & exa-scale data sets
Obsolete Infrastructures

- Data management infrastructures (DBMSs) were not prepared for this deluge

One application
One data type
One user
One CPU

→ Infrastructures collapse
Why is evolving DBMSs so hard?

- Two fundamental problems to tackle
  - Obsolete physical model (impedance mismatch)
    - N-ary storage, relational tuples
    - B/R-trees
    - SPJ queries
      - Catastrophic performance in new application domains
  - Impractical logical guarantees
    - ACID properties
    - Brewer’s conjecture (CAP theorem)
      - Inherent difficulty to scale-out
New Era of Data Management

• Users have gone away of DBMSs...
  ■ File-centric solutions
  ■ Procedural processing chains

• ... and have come back
  ■ Data abstractions (physical & logical)
  ■ Strong consistency guarantees
XLDB Workshops

- Extremely Large Databases Workshops
  - practical issues related to extremely large databases (1PB+)
    - focus on eScience: radio-astronomy, geoscience, genomics, particle physics
  - invitation-only
  - organized since 2007 by Becla & Lim (SLAC)

- Invited database researchers from the start
  - consensus on array data model
  - SciDB was born
    - Open-source, array-oriented database system
    - CIDR paper 2009 [S09]
The SciDB Consortium

- Distributed R&D Team

- Scientific Advisory Board

- Start-up (Zetics)
  - Marylin Matz (CEO) & Paul Brown (CTO)

- Beta version demoed at VLDB 2009 [CM09]
  - First public release (V0.5) available now
II. System Overview
SciDB Philosophy

- Array-Oriented
- Declarative
- Extensible
- Shared-Nothing
- Open-Source
High-Level Architecture

- SciDB Application
  - Language Specific UI
  - Runtime Supervisor
  - Query Interface and Parser
  - Plan Generator
  - Local Executor
  - Storage Manager

Application Layer

Server Layer

Storage Layer
Data Model

- Nested, multidimensional arrays as first-class citizens
  - Basic data types
  - User-Defined Types

![Nested, multidimensional array diagram]
Storage Model (1/3)

- Vertical Partitioning

\[ a_1 = 2 \]
\[ a_2 = 3.2f \]
\[ a_3 = \]

\[(0,0)\]  
\[(5,2)\]  

\(a_1\)  
\(a_2\)  
\(a_3\)
Storage Model (2/3)

- Co-location of values
  - Adaptive chunking
  - Dense-packing
  - Chunks co-location
  - Compression

- Two on-disk representations
  - Dense arrays
    - extremely compact, no position, offsetting
  - Sparse arrays
    - compactly stores positions + values in array order
Storage Model (3/3)

- Co-location of nested arrays
  - Nested array of rank M from parent array of rank N stored as array of rank (M+N)

- Redundancy
  - Array replication
  - Chunking w/ overlap ➔ Parallelism
Operators (1/2)

- Declarative system supporting an array-oriented query language (AQL)

- Three types of operators
  - Structural (subsample, add dimension, concatenate etc.)
  - Content-dependent (filter, select etc.)
  - Both structural & content-dependent (joins)
Operators (2/2)

- Scatter/gather
  - Generic operator to distribute & aggregate array chunks & values in distributed settings
CREATE ARRAY Test_Array
< A: integer NULLS,
  B: double,
  C: USERDEFINEDTYPE >
[ I=0:99999,1000,10, J=0:99999,1000,10 ]
PARTITION OVER ( Node1, Node2, Node3 )
USING block_cyclic();
The 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

Monday, November 29, 2010
User-Defined Functions (UDFs)

- Extensibility through Postgres-style UDFs
  - Encodes all domain-specific operations
  - Coded in C++
  - Logical definition
    - 1-N arrays-in, 1-N arrays-out
- Physical implementation
  - chunk-granularity
  - Receives chunks from local query executor
  - Creates cell iterators, consumes values
  - Returns chunks to local executor
Parallel UDFs

Regrid 2::1 UDF

Worker 1

Worker 2

input pipeline

output pipeline

(16,0)

(12,0)

(8,0)

(4,0)

(2,0)

(0,0)
Demo

- preview of V0.75
- 4 node installation
Optimal Chunking

- How to chunk and index very large arrays?
  - Chunk sizes & index size directly impact all array operations
    - most operations highly IO-bound

- Multifaceted problem
  - Chunks must be relatively large to amortize disk seeks, but not too large to fit in main memory and to avoid reading unnecessary data
  - Data can be highly skewed
  - Queries can be highly skewed
  - Index must be compact and easily updatable
• Cost-based, optimal spatial partitioning
  - Efficient, hierarchical partitioning
Spatial Partitioning (2/2)

- Basic idea
  - **Cost-model** for query execution times based on #cells accessed
  - Optimal quadtree construction based on cost-model, query workload, local density & page size
    - \( \text{cellSize}_{\text{opt}}(Q, D, \text{pageSize}) \) [CM10a] \((\text{TrajStore})\)

- Optimal balance between
  - **Oversized** cells
    - potentially retrieves data that is not queried
  - **Undersized** cells
    - seek not amortized if too little data read
    - unnecessary seeks if dense data and relatively large query
Versioning

- N-dimensional array objects
- Frequent insertions of new versions
  - New observations
  - New simulations
- Arbitrary queries over version history
Problems w/ Versioning Systems

- Current Versioning Systems (e.g., SVN, GIT) are
  - Slow for large / numerous binary files
  - Catastrophic for spatial queries over series of versions
    - No spatial chunking
    - No co-location of versions
AVS Algorithms

- Given a workload of frequent / typical queries:
  - Optimal co-location of (N+1) dimensional chunks on disk
  - Optimal materialization / \(\Delta\)-compression of versions to minimize
    - Storage space
    - Query execution time

\(\Rightarrow\) Orders of magnitude faster than svn / git on scientific workloads
The SS-DB Benchmark

• Three main ways of building a scientific DBMS
  ■ array simulation layer on top of relational DBMS
    ■ MonetDB
  ■ array executor on top of blobs
    ■ Rasdaman
  ■ native array system
    ■ SciDB

• SS-DB [CM10b]
  ■ Suggest realistic workload for eScience
    ■ Queries on raw data, derived data and ensemble data
  ■ Understand the performance differences bw models
Contents

• Receive raw imagery data + metadata
  ■ 2D images
• Load dataset
• Cook the data to find observations
  ■ stars
• Group observations
  ■ follow stars over space & time
• Execute a few queries
  ■ on raw data as well as derived data
Query Workload

- Q1: Aggregation
- Q2: Recooking
- Q3: Regridding
- Q4: Aggregation
- Q5: Polygons
- Q6: Density
- Q7: Group Centroids
- Q8 & Q9: Trajectories
# Current Results

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Dataset</th>
<th>Loading/Cooking [min]</th>
<th>Query Runtimes [min]</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Load</td>
<td>Obsv</td>
<td>Group</td>
</tr>
<tr>
<td>MySQL</td>
<td>small</td>
<td>760</td>
<td>110</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>normal (scaleup)</td>
<td>770</td>
<td>200</td>
<td>90</td>
</tr>
<tr>
<td>SciDB</td>
<td>small</td>
<td>34</td>
<td>1.6</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>normal (scaleup)</td>
<td>67</td>
<td>1.9</td>
<td>1.5</td>
</tr>
<tr>
<td>(MySQL/SciDB)</td>
<td>small</td>
<td>(22)</td>
<td>(69)</td>
<td>(3.3)</td>
</tr>
<tr>
<td></td>
<td>normal</td>
<td>(12)</td>
<td>(105)</td>
<td>(6)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
<th>Q5</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>speedup</td>
<td>time</td>
<td>speedup</td>
</tr>
<tr>
<td>Baseline</td>
<td>120</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>Vertical Stor.</td>
<td>14</td>
<td>8.6</td>
<td>0.7</td>
</tr>
<tr>
<td>Compression</td>
<td>9</td>
<td>13.3</td>
<td>0.4</td>
</tr>
<tr>
<td>Parallelism</td>
<td>8.2</td>
<td>14.6</td>
<td>0.2</td>
</tr>
</tbody>
</table>
SciDB Roadmap

• R0.5 available: very early stage “work-in-progress”
  ■ Basic functionality
  ■ iquery interpreter
  ■ Internal query language - AIL
  ■ Small set of operators - create, aggregate, join, filter, subsample
  ■ Minimal ‘wiki-style’ documentation
  ■ Breathes, crawls, hiccups

• R0.75 targeted for end-of-year
  ■ AQL - the SQL-like array query language
  ■ Error handling
  ■ Scalable math operations
  ■ Better documentation & stability

• R1.0 beta in April 2011
  ■ More functionally complete (UDFs, uncertainty, provenance et al)
  ■ Robust, high performance
Research Agenda for XI

• Array Versioning
  ■ Tests on very large data sets
  ■ Forks
  ■ Distributed versioning

• Data Lineage (Semantic Information)
  ■ Arbitrary information stored in a triple-store
  ■ Integration / performance?
  ■ Automatic creation of workflow metadata
  ■ Arbitrary AQL queries using metadata
References

- SciDB website: http://scidb.org/
- [S09] Requirements for Science Data Bases and SciDB. Stonebraker, Becla, DeWitt, Lim, Maier, Ratzesberger, Zdonik. CIDR. 2009.