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  

\[(A::INTEGER, B::FLOAT) \rightarrow (2, 0.7)\]
Array Definition

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

<table>
<thead>
<tr>
<th>I = 0 : 2</th>
<th>J = 0 : 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2, 0.7)</td>
<td>(2, 0.7)</td>
</tr>
<tr>
<td>(1, 0.5)</td>
<td>(1, 0.5)</td>
</tr>
<tr>
<td>(3, 0.1)</td>
<td>(3, 0.1)</td>
</tr>
<tr>
<td>(6, 0.3)</td>
<td>(6, 0.3)</td>
</tr>
<tr>
<td>(7, 0.1)</td>
<td>(7, 0.1)</td>
</tr>
<tr>
<td>(0, 0.1)</td>
<td>(0, 0.1)</td>
</tr>
<tr>
<td>(1, 0.4)</td>
<td>(1, 0.4)</td>
</tr>
<tr>
<td>(4, 0.6)</td>
<td>(4, 0.6)</td>
</tr>
<tr>
<td>(8, 0.1)</td>
<td>(8, 0.1)</td>
</tr>
</tbody>
</table>

Empty cells

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

• Declarative query language

Slice (Example, I=2);

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

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

• Manipulate data content

\[
\text{Filter (Example, } A > 2)\text{;}
\]

• Operations are \textit{composable}

\[
\text{Apply (Slice (Example, } I = 2\text{), } A * B)\text{;}
\]

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

<table>
<thead>
<tr>
<th>(3, 0.1)</th>
<th>3</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0.1)</td>
<td>0</td>
<td>0.1</td>
</tr>
</tbody>
</table>
2. **Chunking:**
   - Operate on a small area
   - Split into equal-sized (and overlapping) chunks
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