MapReduce and Parallel DBMSs:  
A Comparison of Approaches to Large-Scale Data Analysis

Andrew Pavlo  
University of Maryland – College Park  
September 3, 2009
Co-Authors

- Daniel Abadi (Yale)
- David DeWitt (Microsoft)
- Samuel Madden (MIT)
- Erik Paulson (Wisconsin)
- Alexander Rasin (Brown)
- Michael Stonebraker (MIT)
Today’s Talk

- SIGMOD '09
  - A Comparison of Approaches to Large-Scale Data Analysis

- CACM '09 (submitted)
  - MapReduce and Parallel DBMSs: Friends or Foes?
  - Compare/Contrast with Jeff Dean (Google)
In the beginning...

- DeWitt + Stonebraker Article
  - MapReduce: A Major Step Backwards [1]

MapReduce and Databases

- Understand loading and execution behaviors for common processing tasks.

- Large-scale data access (>1TB):
  - *Analytical query workloads*
  - *Bulk loads*
  - *Non-transactional*
Outline

- MapReduce/DBMS Overview
- Benchmark Study
- Results Analysis & Discussion
- Google’s Response
- Sweet Spots
- Concluding Remarks
MapReduce Overview

- Massively parallel data processing
  - Programming Model vs. Execution Platform

- Programs consist of only two functions:
  - \( \text{Map}(k1, v1) \rightarrow (k2, \text{list}(v2)) \)
  - \( \text{Reduce}(k2, \text{list}(v2)) \rightarrow (\text{key3}, \text{list}(v3)) \)
MapReduce Example

- Calculate total order amount per day.
Shared-Nothing Parallel Databases

- **Common characteristics:**
  - *Data partitioning.*
  - *Inter- and intra-query parallelism.*

- **Modern systems are based on pioneering work from 1980s:**
  - *TeraData (‘86)*
  - *Gamma (DeWitt ‘86)*
  - *Grace (Fushimi ‘86)*
Benchmark Environment

- **Tested Systems:**
  - Hadoop (MapReduce)
  - Vertica (Column-store DBMS)
  - DBMS-X (Row-store DBMS)
- 100-node cluster at Wisconsin.
- Additional configuration information is available on our website.
Methodology

- Report load & execution times.
  - *All results are an average of three trials.*
  - *Flush caches to ensure cold start.*

- Hadoop results include separate combine task to consolidate results on a single-node.
  - *Numbers are reported separately.*
Grep Task

- Find 3-byte pattern in 100-byte record
  - 1 match per 10,000 records

- Data set:
  - 10-byte unique key, 90-byte value
  - 1TB spread across 25, 50, or 100 nodes
  - 10 billion records

- Original MR Paper (Dean et al. 2004)
Grep Task Loading Results

- Hadoop
- Vertica
- DBMS-X

Bar chart showing comparison of loading times for Hadoop, Vertica, and DBMS-X across data sizes 25x40GB, 50x20GB, and 100x10GB.
Grep Task Execution Results

- Hadoop
- Vertica
- DBMS-X

Bar chart showing execution times for 25x40GB, 50x20GB, and 100x10GB datasets.
Analytical Tasks

- Simple web processing schema

- Data set:
  - 600k HTML Documents (6GB/node)
  - 155 million UserVisit records (20GB/node)
  - 18 million Rankings records (1GB/node)
Aggregate Task

- Simple query to find adRevenue by IP prefix

```sql
SELECT SUBSTR(sourceIP, 1, 7), SUM(adRevenue)
FROM userVistits
GROUP BY SUBSTR(sourceIP, 1, 7)
```
Aggregate Task Results

- Hadoop
- Vertica
- DBMS-X

25 nodes: [Bar chart]
50 nodes: [Bar chart]
100 nodes: [Bar chart]
Join Task

- Find the sourceIP that generated the most adRevenue along with its average pageRank.

- Implementations:
  - DBMSs – Complex SQL using temporary table.
  - MapReduce – Three separate MR programs.
Join Task Results

- **Hadoop**
- **Vertica**
- **DBMS-X**

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Hadoop</th>
<th>Vertica</th>
<th>DBMS-X</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>32.0</td>
<td>29.2</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>35.4</td>
<td>29.4</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>55.0</td>
<td>31.9</td>
<td></td>
</tr>
</tbody>
</table>
UDF Task

- First phase of PageRank Algorithm
  - *Count number of links for each URL.*

- DBMS Troubles:
  - *Vertica did not support UDFs.*
  - *DBMS-X had buggy BLOBs.*

- Hadoop implementation is straightforward.
UDF Task Results

- Hadoop
- Vertica
- DBMS-X
Outline

- MapReduce/DBMS Overview
- Benchmark Study
- Results Analysis & Discussion
- Google’s Response
- Sweet Spots
- Concluding Remarks
Implementation Refinement

Aggregation Task (50 nodes)
Task Start-up

- Hadoop is slow to start executing programs:
  - 10 seconds until first Map starts.
  - 25 seconds until all 100 nodes are executing.
  - 7 buffer copies per record before reaching Map function [1].

- Parallel DBMSs are always “warm”

Repetitive Data Parsing

- Hadoop has to parse/cast values every time:
  - *SequenceFiles provide serialized key/value.*
  - *Multi-attribute values must still handled by user code.*

- DBMSs parse records at load time:
  - *Allows for efficient storage and retrieval.*
Outline

- MapReduce/DBMS Overview
- Benchmark Study
- Results Analysis & Discussion
- Google’s Response
- Sweet Spots
- Concluding Remarks
Google’s Response

- Jeffrey Dean and Sanjay Ghemawat
  - *MapReduce: A Flexible Data Processing Tool CACM’09*

- Key points:
  - *Flaws in benchmark.*
  - *Fault-tolerance in large clusters.*
  - *MapReduce ≠ DBMS*
Google’s Response: Flaws

- MR can load and execute queries in the same time that it takes DBMS-X just to load.

- Alternatives to reading all of the input data:
  - Select files based on naming convention.
  - Use alternative storage (BigTable).

- Combining final reduce output.
Google’s Response: Cluster Size

- Largest known database installations:
  - *Greenplum* – 96 nodes – 4.5 PB (eBay) [1]
  - *Teradata* – 72 nodes – 2+ PB (eBay) [1]

- Largest known MR installations:
  - *Hadoop* – 3658 nodes – 1 PB (Yahoo) [2]
  - *Hive* – 600+ nodes – 2.5 PB (Facebook) [3]

[1] eBay’s two enormous data warehouses – April 30\textsuperscript{th}, 2009
  http://developer.yahoo.net/blogs/hadoop/2009/05/hadoop_sorts_a_petabyte_in_162.html
Google’s Response: Functionality

- MapReduce enables parallel computations not easily performed in a DBMS:
  - *Stitching satellite images for Google Earth.*
  - *Generating inverted index for Google Search.*
  - *Processing road segments for Google Maps.*

- Programming Model vs. Execution Platform
Outline

- MapReduce/DBMS Overview
- Benchmark Study
- Results Analysis & Discussion
- Google’s Response
- Sweet Spots
- Concluding Remarks
Extract-Transform-Load

- “Read Once” data sets:
  - Read data from several different sources.
  - Parse and clean.
  - Perform complex transformations.
  - Decide what attribute data to store.
  - Load the information into a DBMS.

- Allows for quick-and-dirty data analysis.
Semi-Structured Data

- MapReduce systems can easily store semi-structured data since no schema is needed:
  - *Typically key/value records with a varying number of attributes.*

- Awkward to store in relational DBMS:
  - *Wide-tables with many nullable attributes.*
  - *Column store fairs better.*
Limited Budget Operations

- MapReduce frameworks:
  - Community supported and driven.
  - Attractive for projects with modest budgets and requirements.

- Parallel DBMSs are expensive:
  - No open-source option.
Concluding Remarks

- What can MapReduce learn from Databases?
  - Declarative languages are a good thing.
  - Schemas are important.

- What can Databases learn from MapReduce?
  - Query fault-tolerance.
  - Support for in situ data.
  - Embrace open-source.
Other Benchmarked Systems

- HadoopDB (Abadi ’09 - Yale)
  - Replaced Hadoop filesystem with Postgres.
  - Makes JDBC calls inside of MR functions.

- Hive (Thusoo ’09 - Facebook)
  - Data warehouse interface on top of Hadoop.
  - Converts SQL-like language to MR programs.
Conclusion

- MapReduce goodness:
  - *Ease of use, “out of box” experience.*
  - *Attractive fault tolerance properties.*
  - *Fast load times.*

- Database goodness:
  - *Fast query times.*
  - *Schemas.*
  - *Supporting tools.*
More Information

- Complete benchmark information and source code is available at our website:
  - http://database.cs.brown.edu/sigmod09/

- Questions/Comments?