A Comparison of Approaches to Large-Scale Data

Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel Abadi, David DeWitt, Samuel Madden, and Michael Stonebraker

SIGMOD'09

Presented by: Daniel Isaacs

Introduction

- It all starts with cluster computing.
- MapReduce
- Why not use a parallel DBMS instead?
- The goal is to understand the differences between MapReduce and parallel DBMS systems to perform large-scale data analysis.

Two Approaches

- Both run on a “shared nothing” collection of computers.
- Parallelism.
- MapReduce: Map and Reduce functions.
- Parallel DBMS: Uses an optimizer that translates SQL commands into a query plan, whose execution is divided amongst multiple nodes.

Architectural Elements

1. Schema Support

- Parallel DBMS require data to fit into rows and columns paradigm.
- MapReduce does not require a structure to their data.
- DBMS separates schema from the application. MapReduce forces programmers to agree on a single schema.

Architectural Elements

2. Indexing

- DBMS use hash or B-tree.
- MapReduce does not provide built-in indexes.

Architectural Elements

3. Programming Model

- Relational vs Codasyl
- Stating what you want or presenting an algorithm for how to get it.
- DBMS (Relational) vs MapReduce (Codasyl)

Architectural Elements

4. Data Distribution

- Parallel DBMS use a query optimizer to balance computational workloads while minimizing the amount of data transfer over a network.
- MapReduce imports a large amount of data from the node and then filters it, generating more network I/O

Architectural Elements`

5. Execution Strategy

- Pull (MapReduce) vs Push (DBMS)
- Example: if N (map instances) is 1000 produces M (output files) 500, the map phase produces 500,000 local files.
- Reduce phase starts, each instance needs to read its 1000 input files and use a FTP to “pull” the input files from the nodes.
- Disk seeks and slowing the disk transfer rate

Architectural Elements

6. Flexibility

- MapReduce supporters argue that DBMS technologies are not flexible.
- Nowadays, DBMS products support user-defined functions, store procedures, and user-defined aggregates.

Architectural Elements

7. Fault Tolerance

- MapReduce provides a more sophisticated failure mechanism that DBMS.
- DBMS transactions need to be restarted in an event of failure.

Performance Benchmarks Environment

Hadoop:

- Version 0.19.0 running on Java 1.6.0
- 256 MB data blocks
- 512 MB heap size for the Task executor JVM
- 1024 MB DataNode/JobTracker
- Enabled rack awareness.

Performance Benchmarks Environment

DBMS-X:

- Latest version of a major relational database vendor.
- Stores data in a row-based format.
- Enabled data compression option.

Vertica:

- Stores data in column-based format.
- Compresses data by default.

Grep Task

- Each system must scan a dataset of 100 bytes records looking for a three character pattern.
- 100 bytes = 10 bytes (a unique key) + 90 bytes (random values)
- Two different datasets: 535MB data per node, and 1TB of data

Grep Task

Data Loading

- Hadoop: command-line utility to upload files into HDFS (Hadoop Distributed File System).
- DBMS-X: two steps. First, they executed a LOAD SQL command. Then an administrative command to reorganize the data on each node.
- Vertica: COPY SQL command.

Grep Task

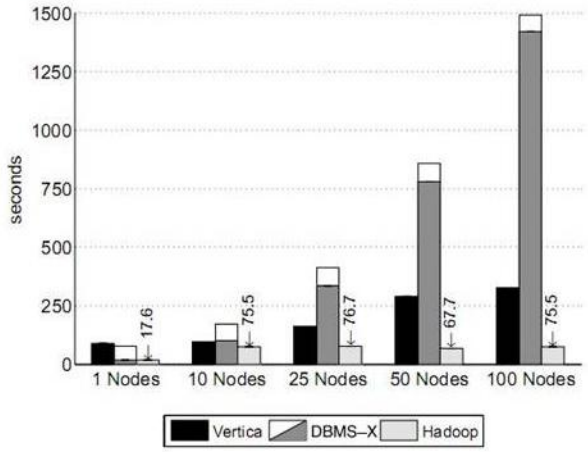

Figure 1: Load Times – Grep Task Data Set (535MB/node)

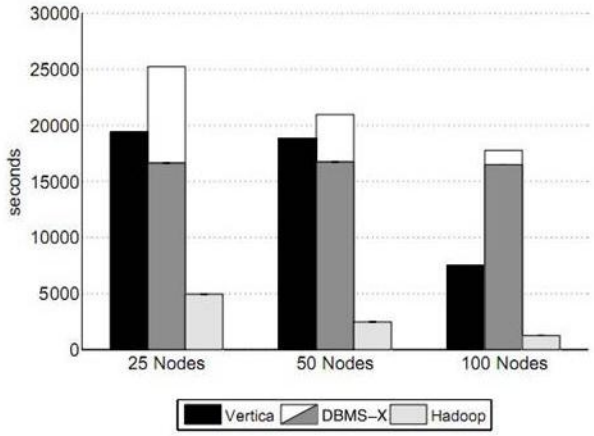

Figure 2: Load Times – Grep Task Data Set (1TB/cluster)

Results:

- Difference in performance between DBMS-X and Vertica and Hadoop.
- Hadoop outperforms both DBMS.

Grep Task

Pattern search

- SELECT * FROM Data WHERE field LIKE '%XYZ%'
- MapReduce only needs a Map function that is given a single record, split into key/value pairs and then performs a sub-string match on the value.

Grep Task

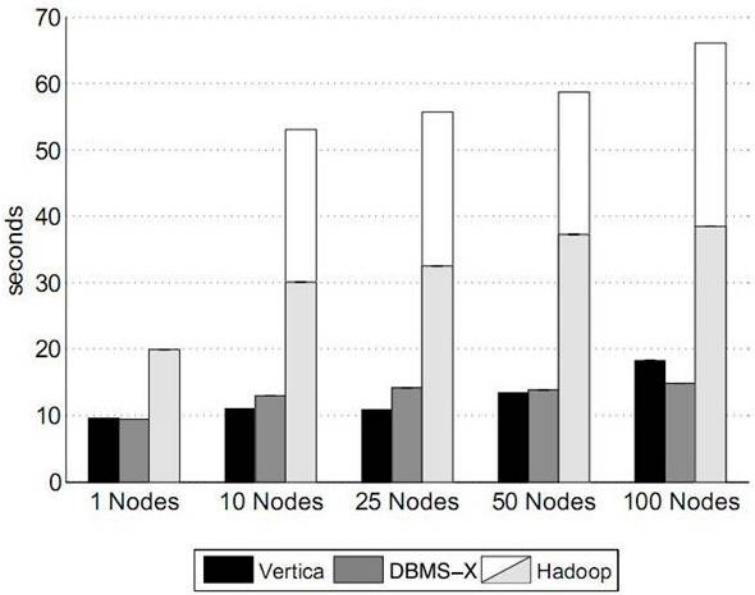

Figure 4: Grep Task Results – 535MB/node Data Set

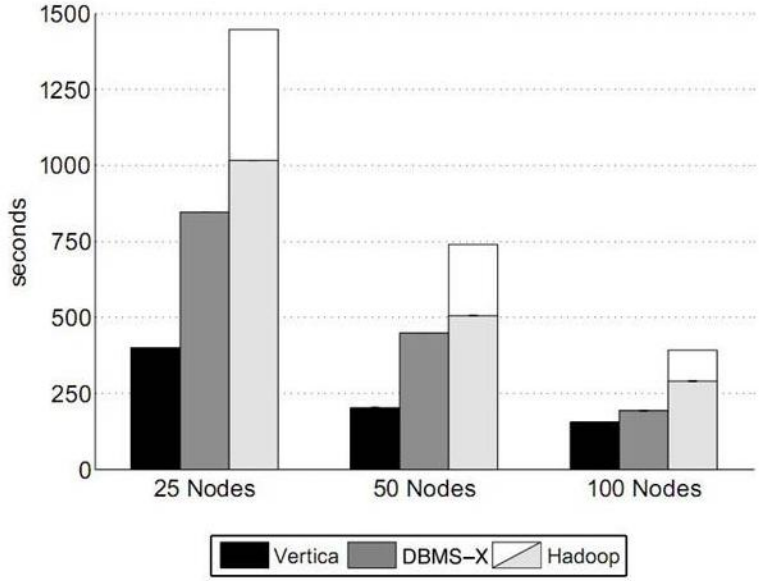

Figure 5: Grep Task Results – 1TB/cluster Data Set

Results:

- In figure 4, both DBMS perform a factor of two faster than Hadoop
- In figure 5, Vertica performs a factor of two faster than Hadoop and DBMS-X

Analytical Task

- To explore more complex uses of Hadoop and DBMS, the authors developed three tables called: Documents, UserVisits and Rankings table.
- Their data generator created 155 million records for UserVisits and 18 million records for the Rankings table.

Analytical Task

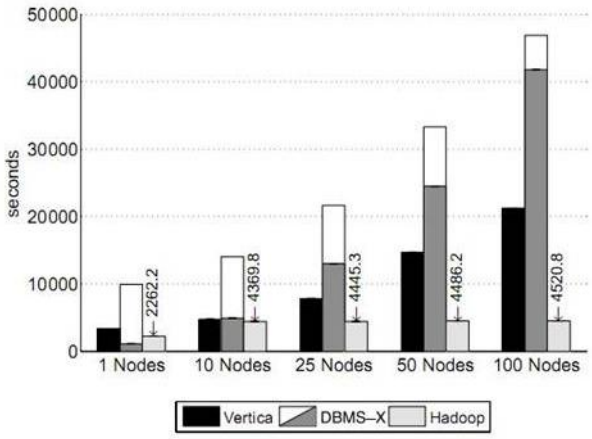

Figure 3: Load Times – UserVisits Data Set (20GB/node)

Data Loading for UserVisits and Rankings

- Hadoop: created a custom data loader executed in parallel on each node to read each line, prepare the data and write tuples into the HDFS.
- DBMS-X and Vertica used the same loading procedures as the Grep Task.

Selection Task

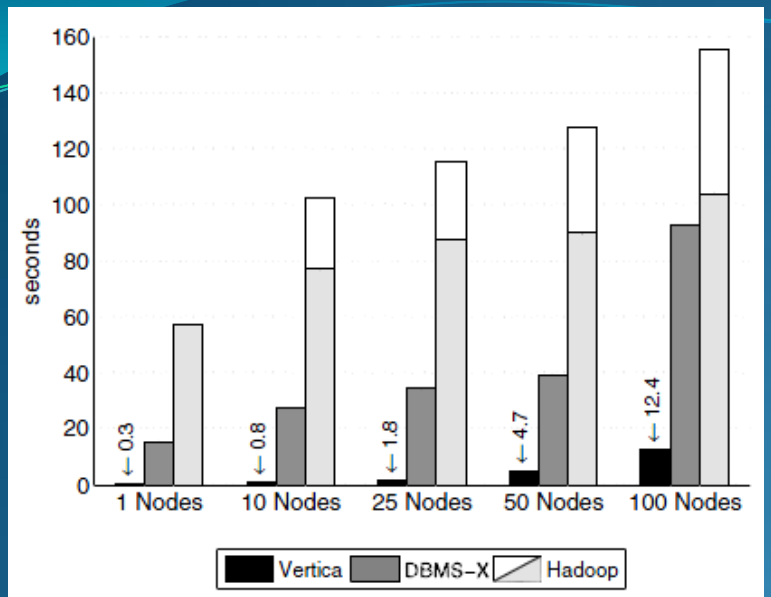

Figure 6: Selection Task Results

SELECT pageURL, pageRank FROM Rankings WHERE pageRank >x

- Hadoop: only a Map Function is required.

Results:

- Parallel DBMSs outperform Hadoop, due to Hadoop's increased start-up costs as more nodes are added to the cluster. Indexes on the pageRank column also improve DBMSs performance.

Aggregation Task

- Each system will calculate the total adRevenue generated for each sourceIP grouped by sourceIP.
- Designed to measure the performance of parallel analytics on a single read-only table.
- MapReduce: requires Map and Reduce functions.
- Map function splits input value by the field delimiter, and outputs the sourceIP field.
- Reduce Function adds together all of the adRevenue values for each sourceIP and then outputs the prefix and revenue total

Aggregation Task

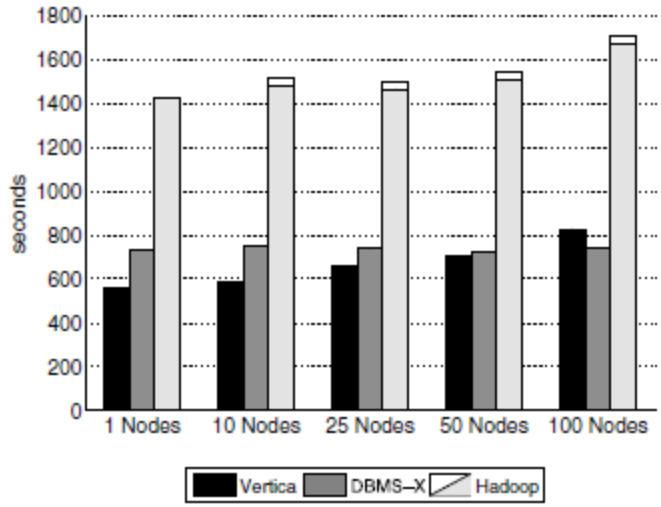

Figure 7: Aggregation Task Results (2.5 million Groups)

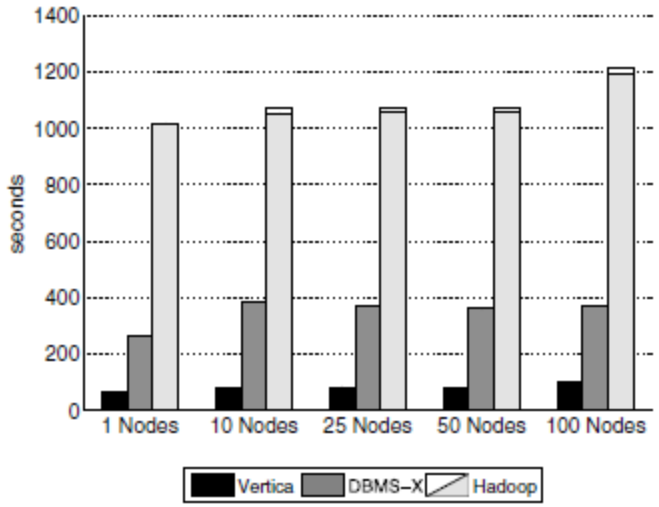

Figure 8: Aggregation Task Results (2,000 Groups)

SELECT sourceIP, SUM(adRevenue)
FROM UserVisits GROUP BY sourceIP;

SELECT SUBSTR(sourceIP, 1, 7), SUM(adRevenue)
FROM UserVisits GROUP BY SUBSTR(sourceIP, 1, 7);

JoinTask

- Combine two data sets and join them together in order to find pairs of Ranking and UserVisits records with matching values for pageURL and destURL.
- DBMS: Two queries. First statement creates a temporary table and uses it to store the output of the SELECT statement that performs the join of the tables and computes the aggregates.
- MapReduce program was broken in three phases.

Join Task

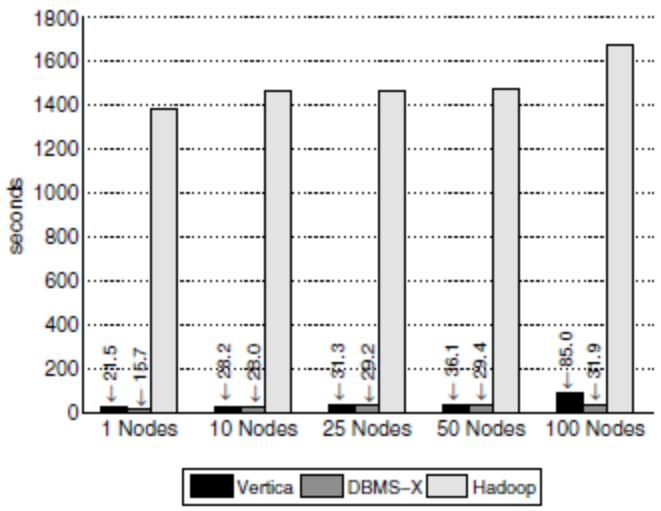

Figure 9: Join Task Results

```
SELECT INTO Temp sourceIP,
AVG(pageRank) as avgPageRank,
SUM(adRevenue) as totalRevenue
FROM Rankings AS R, UserVisits AS UV
WHERE R.pageURL = UV.destURL
AND UV.visitDate BETWEEN Date('2000-01-15')
AND Date('2000-01-22')
GROUP BY UV.sourceIP;

SELECT sourceIP, totalRevenue, avgPageRank
FROM Temp
ORDER BY totalRevenue DESC LIMIT 1;
```

Results

- Big difference between Hadoop and DBMS.
- Hadoop is limited by the speed it can read the UserVisits table off the disk.
- No network overhead for DBMS.

UDF Aggregation Task

- Compute an inlink count for each document in the dataset. Systems must read each document file and search for the URLs that appear in the contents.
- Then the systems, for each unique URL, must count the number of unique pages that reference that particular URL across a set of files.

UDF Aggregation Task

- DBMS-X: modified the UDF to open each HTML document on the local disk and process its contents as if it was stored in the database.
- Vertica: extract URLs from files but write the output to files on each local file system node.
- MapReduce: HTML documents split by lines where the line content is the value and the line number its key. Reduce function counts the number of values for a given key.

UDF Aggregation Task

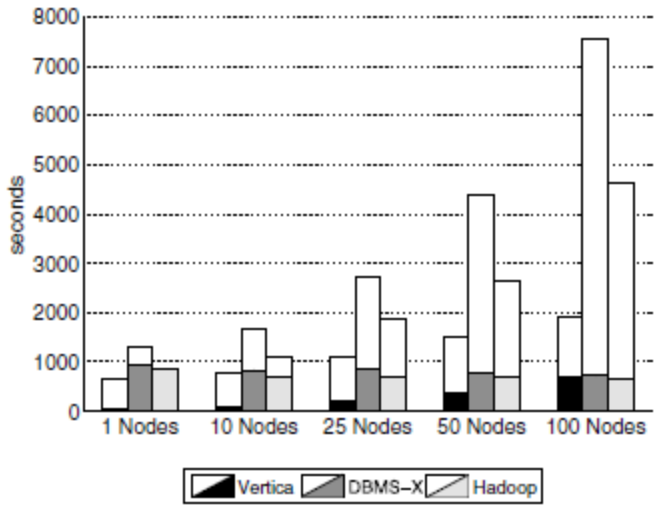

Figure 10: UDF Aggregation Task Results

SELECT INTO Temp F(contents) FROM Documents;

SELECT url, SUM(value) FROM Temp GROUP BY url;

Results

- DBMS-X performs poorly due to the added overhead of row-by-row interaction between UDF and the input file outside of the database.

System-level Aspects

1. System Installation, Configuration and Tuning

- Hadoop was installed with little effort
- DBMS-X installation was straightforward. Nonetheless, configuration was complicated and it was ineffective at adjusting memory allocation.
- Vertica was install as an RPM that was deployed on each node. Default setting worked well.

System-level Aspects

1. System Installation, Configuration and Tuning

- Hadoop was easier to install and configure than DBMS.
- Tuning in DBMS is mostly done prior to query execution.
- DBMS came with tools for tuning, whereas Hadoop was a trial and error tuning.

System-level Aspects

2. Task Start-up

- On a cluster of 100 nodes, Hadoop takes 10 seconds from the moment a job is submitted to the JobTracker, and 25 seconds until all nodes execute the job.
- DBMSs start at OS boot time.

System-level Aspects

3. Compression

- DBMS-X and Vertica allow for compression of stored data.
- Compression in Hadoop either block-level or record-level slowed execution.

System-level Aspects

4. Loading and Data Layout

- DBMSs do the parsing at load time extracting attributes very quickly.
- Hadoop must parse and deserialize the records at run time, becoming more CPU intensive.
- Hadoop Loading process achieved faster load throughputs than Vertica and DBMS-X.
- For Data that only is going to be loaded once, indexing and reorganization is not worth it.

System-level Aspects

5. Execution Strategies

- DBMSs transfers data between nodes only when its necessary. No control messages are used during processing
- Hadoop uses a large number of control messages to synchronize processing, resulting in poor performance due to an increased overhead.

System-level Aspects

6. Failure Model

- MapReduce is able to recover from faults in the middle of query execution.
- DBMSs need to restart the query.

User-level Aspects

1. Ease of Use

- MapReduce framework can check if a user's code compiles, SQL Engines determine whether the queries parse correctly.
- Programmers are more familiar with OOP technologies than languages such as SQL. However SQL has good portability.
- MapReduce is easier to get running, but more expensive to maintain.

User-level Aspects

2. Additional Tools

- Hadoop has a web interface that allows users to browse the contents of the HDFS and monitor job execution
- DBMSs have data visualization tools, business intelligence, data mining, data replication, and others.

Conclusion

- DBMS-X was 3.2 times faster than MapReduce.
- Vertica was 2.3 times faster than MapReduce

DBMS performs better because:

- B-Tree indices to speed the execution of selection operations.
- Novel storage mechanisms (column-orientation)
- Aggressive compression techniques with ability to operate directly on compressed data
- Sophisticated parallel algorithms for querying large amounts of relational data.

Conclusion

Nonetheless, Hadoop has certain advantages:

- Easy to install and use
- Fault Tolerance
- Extensibility