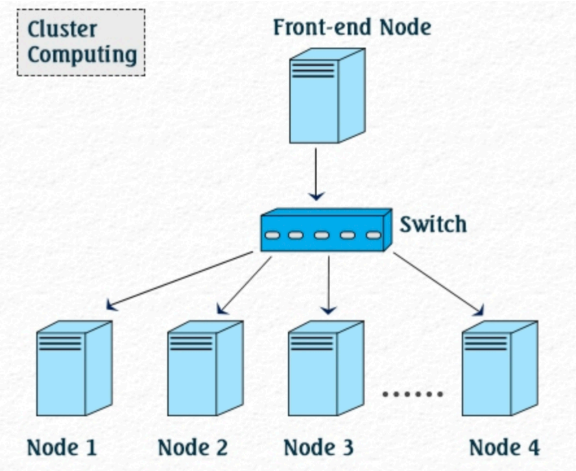
A Comparison of Approaches to Large-Scale Data Analysis

Inder Preet Singh
Machine Learning Researcher
1QBiT

Contents

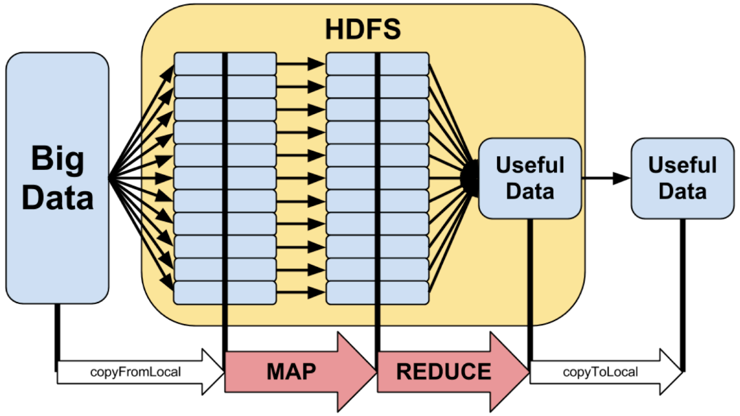
Introduction

- Cluster Computing



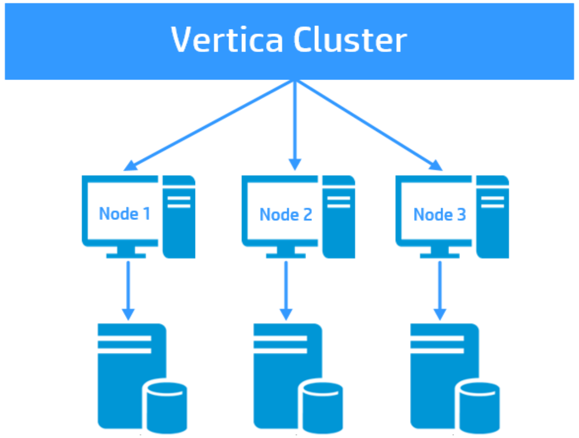
Introduction

- Cluster Computing
- MapReduce



Introduction

- Cluster Computing
- MapReduce
- Parallel DBMS



Two Approaches

- Map Reduce
- Parallel DBMS

Map Reduce

Consists of following phases:

- Map
- Reduce
- MapReduce Scheduler

Map Reduce

Map Reduce Analogy

Parallel DBMS

Two key aspects to enable parallel execution:

- Partioning of Tables
- Optimizer

Parallel DBMS

Working of Parallel DBMS:

- SQL command to filter the records in a table T1 based on a predicate, along with a join to a second table T2 with an aggregated computed on result of join. What if ?
- T2 is small
- T2 is large

Architectural Elements

- Schema Support
- Indexing
- Programming Model
- Data Distribution
- Flexibility
- Fault Tolerance

Architectural Elements: Schema Support

- Parallel DBMS: Relational Schema
- Map Reduce

Can we say that Map Reduce is preferable over Parallel DBMS?

- Complicated Keys
- Sharing of Data

Architectural Elements: Indexing

- Parallel DBMS: Hash or B-tree
- Map Reduce: No built-in indexes

Architectural Elements: Programming Model

How to write program to access the data:

- Relational
- Codassyl

Architectural Elements: Data Distribution

- Parallel DBMS: Query Optimizer
- Map Reduce

Architectural Elements: Flexibility

- Map Reduce
- Parallel DBMS

Architectural Elements: Fault Tolerance

- Map Reduce
- Parallel DBMS

Performance Benchmarks:

- Benchmark Environment
- Benchmark Execution
- The Original MR Task
- Analytical Tasks

Performance Benchmarks: Benchmark Environment

- Hadoop
  - Open Source Implementation of Map Reduce
  - Hadoop Version 0.19.0
- DBMS-X
  - Row Based Format
  - Doesn't compress data
- Vertica
  - Column Based Format
  - Compress data by default

Performance Benchmarks: Benchmark Execution

- Execute each task 3 times.
- Multiple Nodes - Scalability

Performance Benchmarks: The Original MR Task

- "Grep Task": Scan through 100 byte records looking for three byte pattern.
- Hadoop
- DBMS-X and Vertica

Performance Benchmarks: The Original MR Task

- "Grep Task": Scan through 100 byte records looking for three byte pattern.
- Hadoop
- DBMS-X and Vertica

```
Create TABLE Data (
        Key VARCHAR(10) PRIMARY KEY,
        Field VARCHAR(90)   );
```

Performance Benchmarks: The Original MR Task

Execute "Grep Task" on two different data sets.

- Fix the size of data per node
- Fix the total data size

Two task:

- Data Loading
- Task Execution

Performance Benchmarks: The Original MR Task

Data Loading

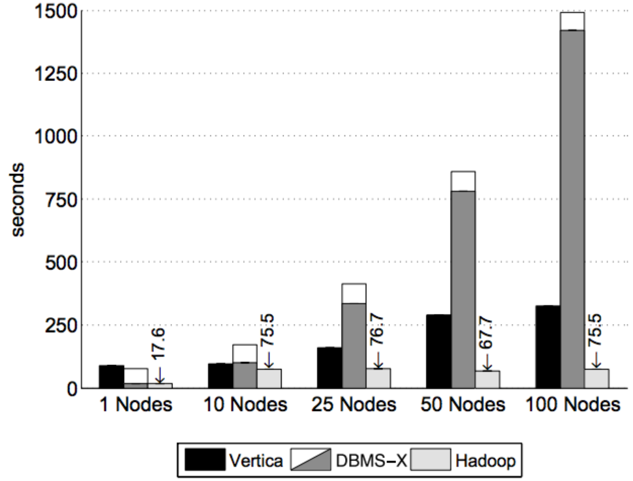


Figure 1: Load Times – Grep Task Data Set (535MB/node)

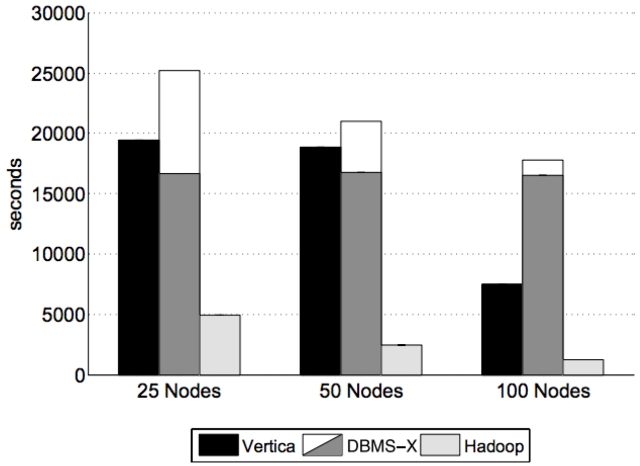


Figure 2: Load Times – Grep Task Data Set (1TB/cluster)

Performance Benchmarks: The Original MR Task

Task Execution

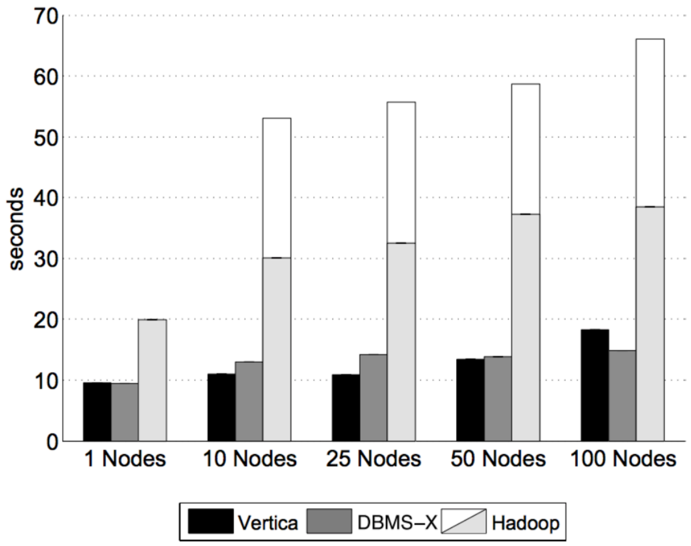


Figure 4: Grep Task Results – 535MB/node Data Set

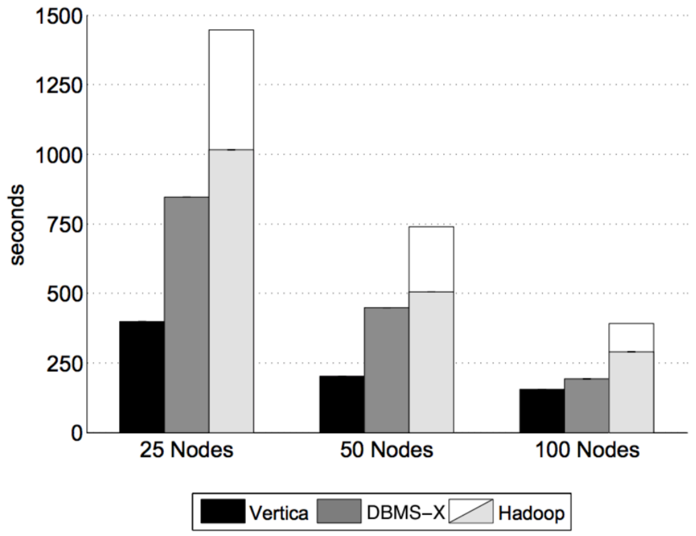


Figure 5: Grep Task Results – 1TB/cluster Data Set

Performance Benchmarks: Analytical Tasks

Developed tasks related to HTML document processing.

Each node is assigned a set of 600,000 unique HTML documents.

Performance Benchmarks: Analytical Tasks

Selection Task

DBMS: Select pageURL, pageRank
FROM Rankings WHERE pageRank > X;

MapReduce: Single Map function, no reduce functionality

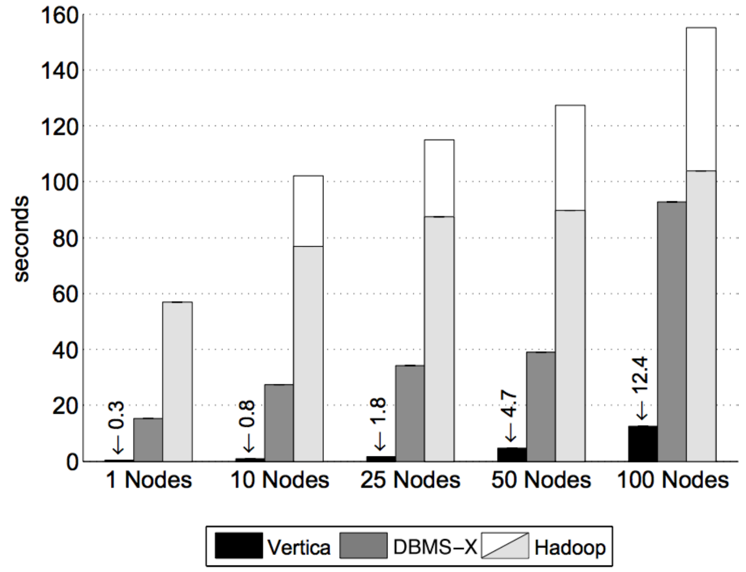


Figure 6: Selection Task Results

Performance Benchmarks: Analytical Tasks

Aggregation Task

DBMS: Select sourceIP, SUM(adRevenue)

FROM UserVisits GROUP BY sourceIP;

MapReduce: Both Map and Reduce functionalities

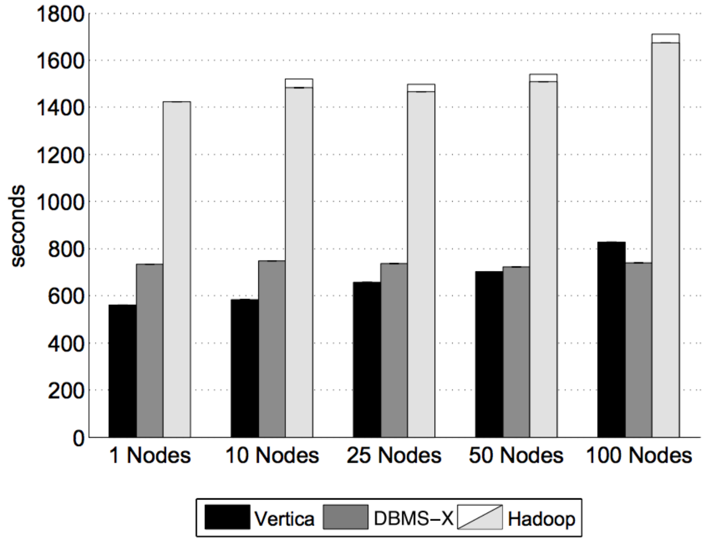


Figure 7: Aggregation Task Results (2.5 million Groups)

Conclusions

- Parallel database systems displayed a significant performance advantage over Hadoop MR in executing a variety of data intensive tasks.
- Parallel systems did not do a good job on the UDF aggregation tasks.
- Parallel systems have less fault tolerance as compared to Hadoop.