A Comparison of Approaches to Large-Scale Data Analysis

SAEED SOLTANI

CMPT 843: TRADITIONAL VS. MODERN DATABASE SYSTEMS

Agenda

Why we need parallel-computing

Two approaches to large scale database systems

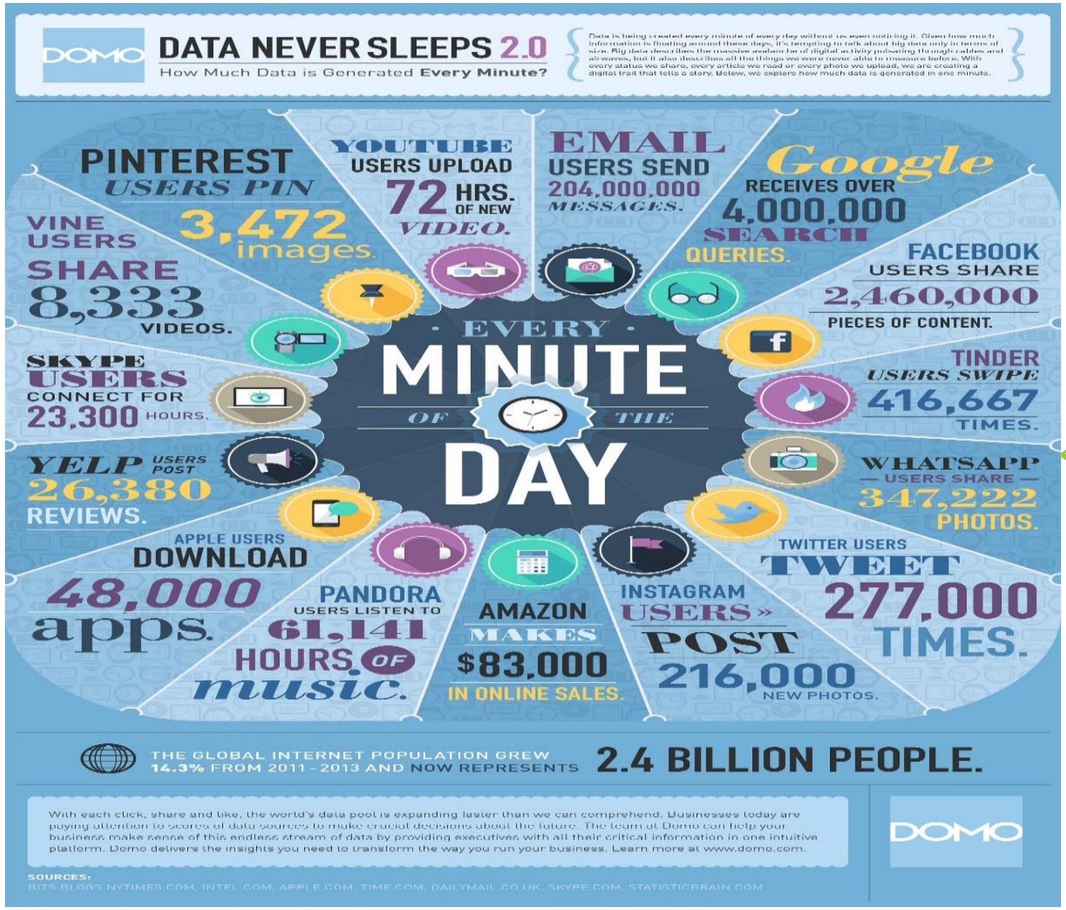
Architectural elements

Performance benchmarks

Discussion

Conclusion

Why do we need Parallel computing?



- Storage
- Analyzing
- Access

https://www.domo.com/blog/data-never-sleeps-2-0/

MapReduce

Started in 2004 by Google

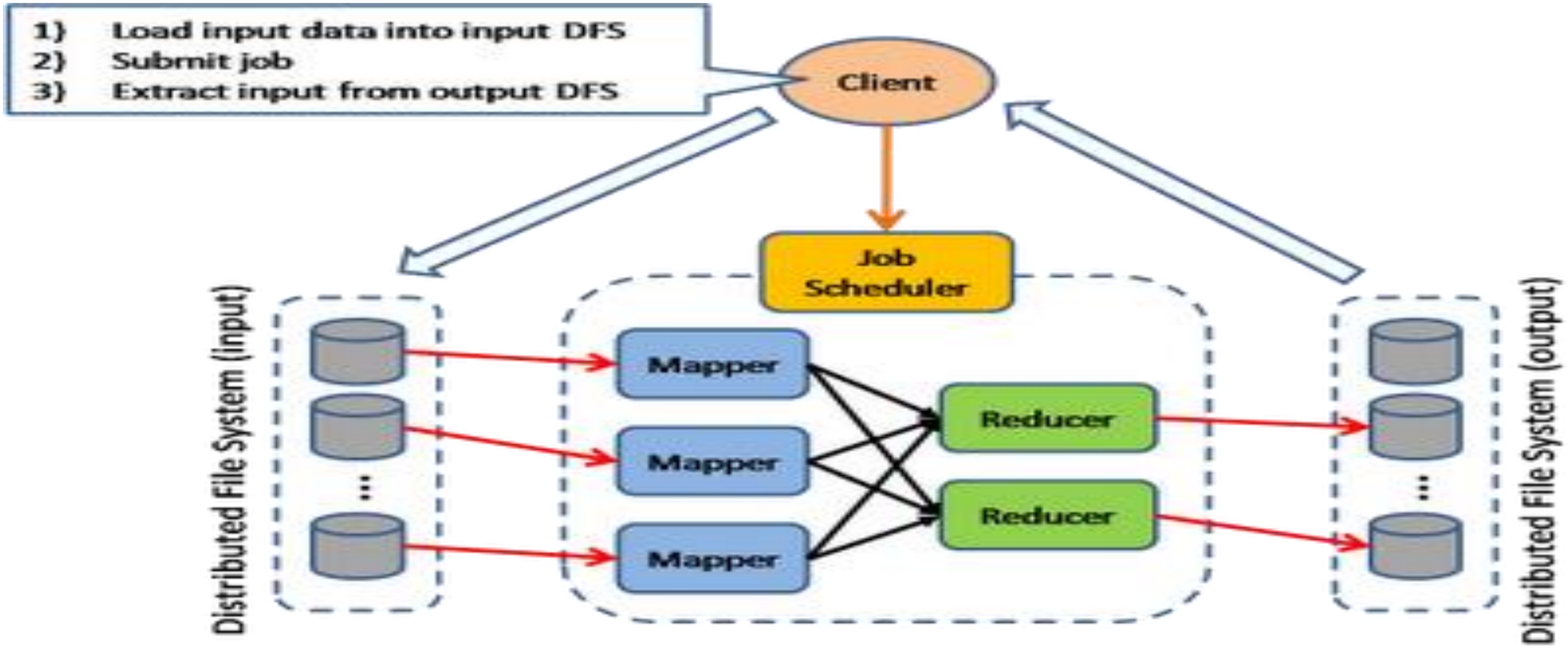
The goal was to solve parallelization, distribution and fault-tolerance

2006, Rise of Hadoop

How MapReduce Works

MR consists of only two functions Map and Reduce

Map used for transformation, Reduce for aggregation

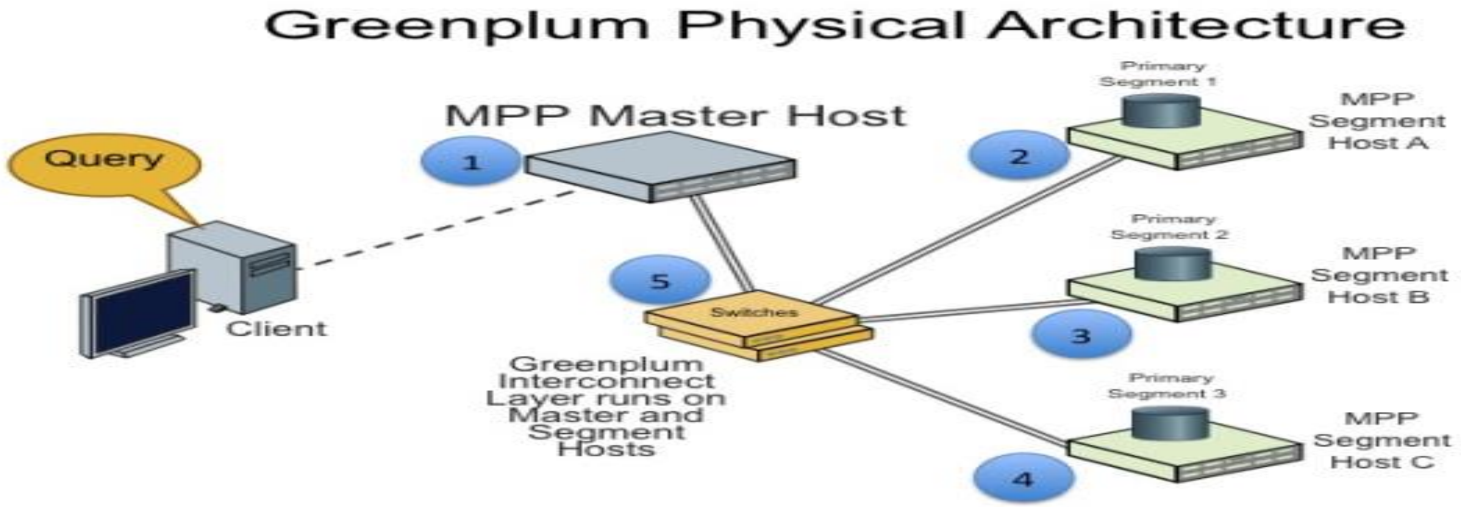


Parallel DBMSs

- Been around since 1980s
- Pioneered by Gamma and Grace
- Support standard tables and SQL

Parallel Databases workflow

- It has four architectures shared memory, shared disk, shared nothing and hierarchical
- data is partitioned across multiple servers or nodes



Architectural Elements

1- Schema Support

Parallel Database VS **MapReduce Model**

Parallel Database

- Accept relational model data
- Data integrity is preserved by system
- Flexible for shared use among users

MapReduce Model

- Accept any type of data
- MR itself does not support Data Integrity
- Suitable when sharing is not priority

Architectural Elements

2- Indexing

Parallel Database VS **MapReduce Model**

Parallel Database

- Use B-tree or hashes for fast access time
- Support multiple indexes per table
- Query optimizer control queries

MapReduce Model

- MR frameworks don’t provide built-in indexes
- Programmer is in charge of implementing index
- Data fetching mechanisms needs tweaking

Architectural Elements

3- Programming Model

Parallel Database VS **MapReduce Model**

Parallel Database

- Relational advocates or CODASYL advocates?
- Relational is stating what you want
- CODASYL is presenting an algorithm for data access

VS

MapReduce Model

- Similar to CODASYL programming model
- Sometimes require low-level programming
- MR community is migrating high-level tasks into runtime

Architectural Elements

4- Data Distribution

Parallel Database VS **MapReduce Model**

Parallel Database

- Query Optimizer balances the workload between nodes and network
- Everything is performed automatically

MapReduce Model

- Map Scheduler schedule Map instances
- Programmer perform data distribution tasks

Architectural Elements

5- Other Elements

Flexibility: **MR** provides more generality, however **DBMS** has added multiple features in the recent years

Fault Tolerance: Both use some replication models. **MR** is more resistant. It can restart the task on a new node in case of failure during the run-time. **DBMS**s avoid saving intermediate results to disk whenever possible

Performance Benchmarks

We compare the two models with testing them against five tasks

Grep Task - scan through a dataset of 100-byte records looking for a three-character pattern.

Analytical Tasks – consist of five tasks related to HTML processing, these tasks are:

1- Selection Task
2- Aggregation Task
3- Join Task
4- UDF aggregation task

Benchmark Environment

Tested Systems

Ver: 0.19.0
Java version: 1.6.0
Default Conf, except
Data Blocks: 256MB
JVM heap size: 512 MB
Data node size: 1024MB
2 mapper,1 reducer each node
HDFS for storage

DBMS-X

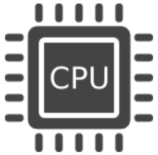
Parallel SQL DBMS
Row-based storage
4GB shared memory
Each table is hash partitioned
Tables are indexed
Tables are compressed
Replication is disabled

Column-based storage
Tables are compressed
Clustered-index indexing
256MB buffer size per node

Benchmark Environment

Node Configuration

✓ 100-node Cluster with Red hat 5, each node:



2.40 GHz Intel Core 2

4GB

250GB SATA-I

1- Grep Task

- ✓ 1TB of data on approximately 1800 files, which is 5.6 million records or roughly 535MB of data per node
- ✓ execute the Grep task on cluster sizes of 1, 10, 25, 50, and 100 nodes
- ✓ We test it with two different datasets
- ✓ Performance shows **scalability** of the system as data grows

Table Structure for parallel DBMS:

```
CREATE TABLE Data (
key VARCHAR(10) PRIMARY KEY,
field VARCHAR(90) );
```

1- Grep Task – Data Loading

- ✓ Hadoop simply outperforms Vertica and DBMS-X
- ✓ In DBMS-X data was loaded sequentially, despite using LOAD command
- ✓ Keep in mind in Hadoop each node performs the replication in parallel
- ✓ Using single replica increases the speed by a factor of 3

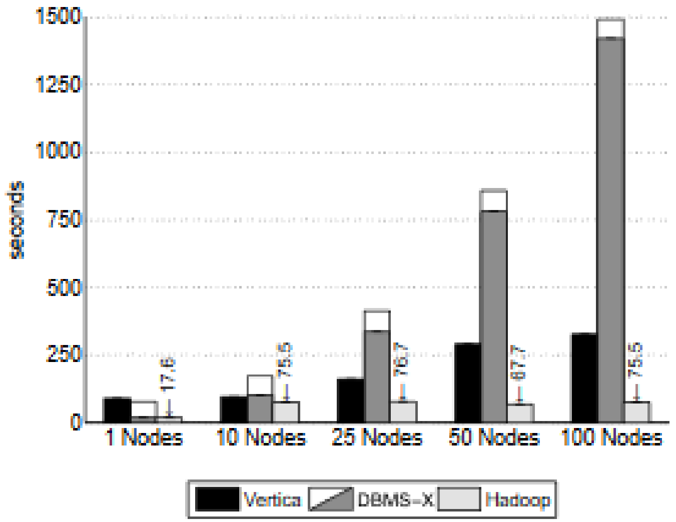


Figure 1: Load Times – Grep Task Data Set (535MB/node)

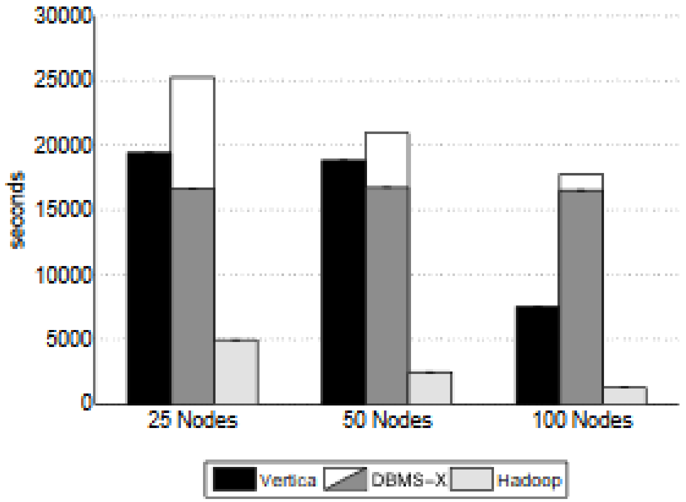


Figure 2: Load Times – Grep Task Data Set (1TB/cluster)

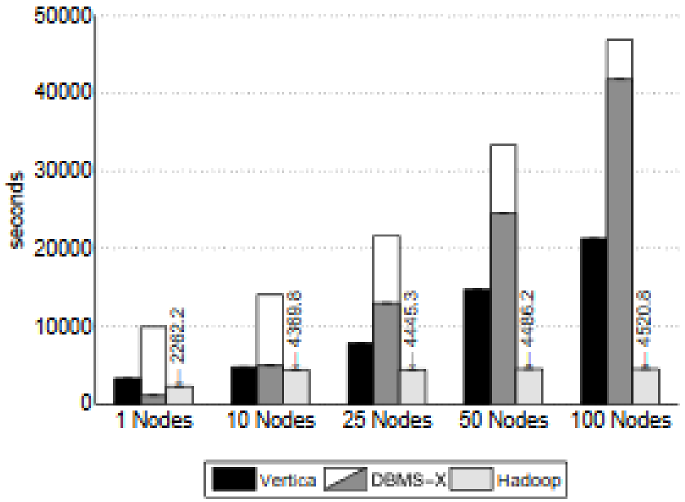


Figure 3: Load Times – UserVisits Data Set (20GB/node)

1- Grep Task – Task Execution

- ✓ **SQL** Command: SELECT * FROM Data WHERE field LIKE '%XYZ%';
- ✓ **MapReduce** program: Map for sub-string search and output the result, No Reducer
- ✓ **Parallel DBMSs** perform more than 2X faster on 535MB/node Data Set
- ✓ Thanks to data compression**, Vertica** is the best one in execution time

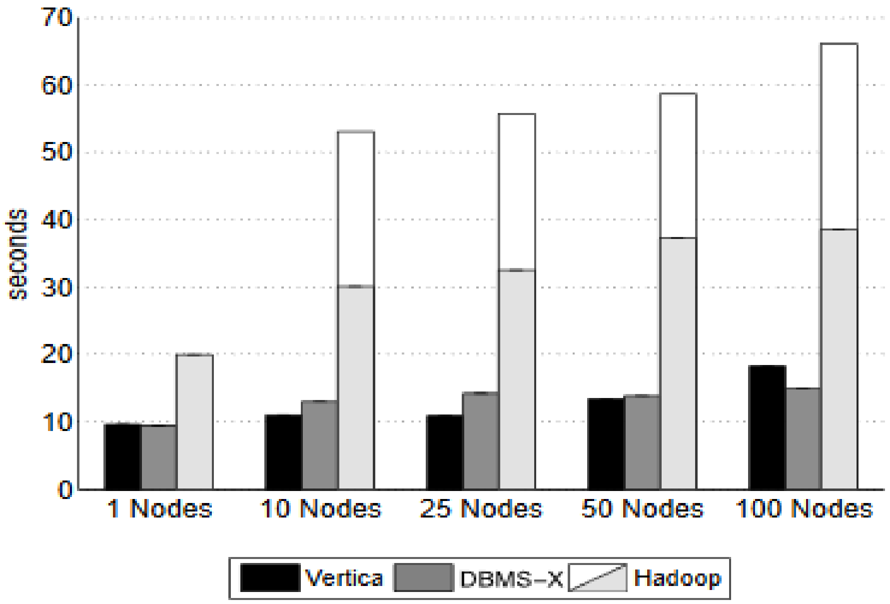


Figure 4: Grep Task Results – 535MB/node Data Set

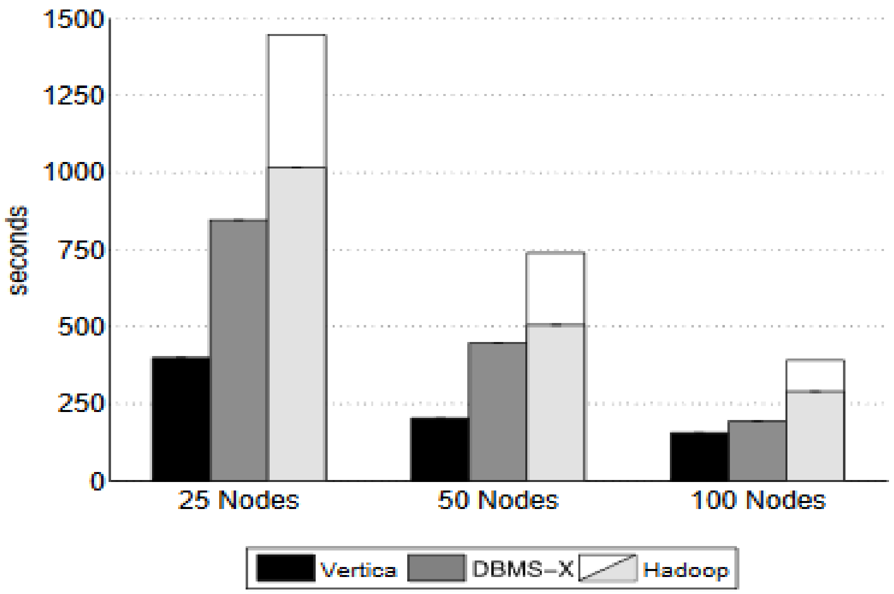


Figure 5: Grep Task Results – 1TB/cluster Data Set

2- Analytical Tasks

Includes four tasks related to HTML processing:

1. Selection Task
2. Aggregation Task
3. Join Task
4. UDF Aggregation Task

Each node has 600,000 unique HTML documents with unique URL with links to other pages

Two other Data sets to model log files of HTTP server traffic.

```
CREATE TABLE Documents (
url VARCHAR(100)
PRIMARY KEY,
contents TEXT );
```

600,000 records

```
CREATE TABLE Rankings (
pageURL VARCHAR(100)
PRIMARY KEY,
pageRank INT,
avgDuration INT );
```

18 million records

```
CREATE TABLE UserVisits (
sourceIP VARCHAR(16),
destURL VARCHAR(100),
visitDate DATE,
adRevenue FLOAT,
userAgent VARCHAR(64),
countryCode VARCHAR(3),
languageCode VARCHAR(6),
searchWord VARCHAR(32),
duration INT );
```

155 million records

2- Analytical Tasks – Selection Task

- ✓ **SQL** Command: SELECT pageURL, pageRank FROM Rankings WHERE pageRank > X;
- ✓ **Parallel DBMSs** outperform MapReduce by a significant factor
- ✓ Hadoop startup costs and indexing the Page Rank by DBMSs are main factors
- ✓ Relative performance degrade with more data and number of nodes
- ✓ **Vertica** reliable message layer causes the overhead with more number of nodes

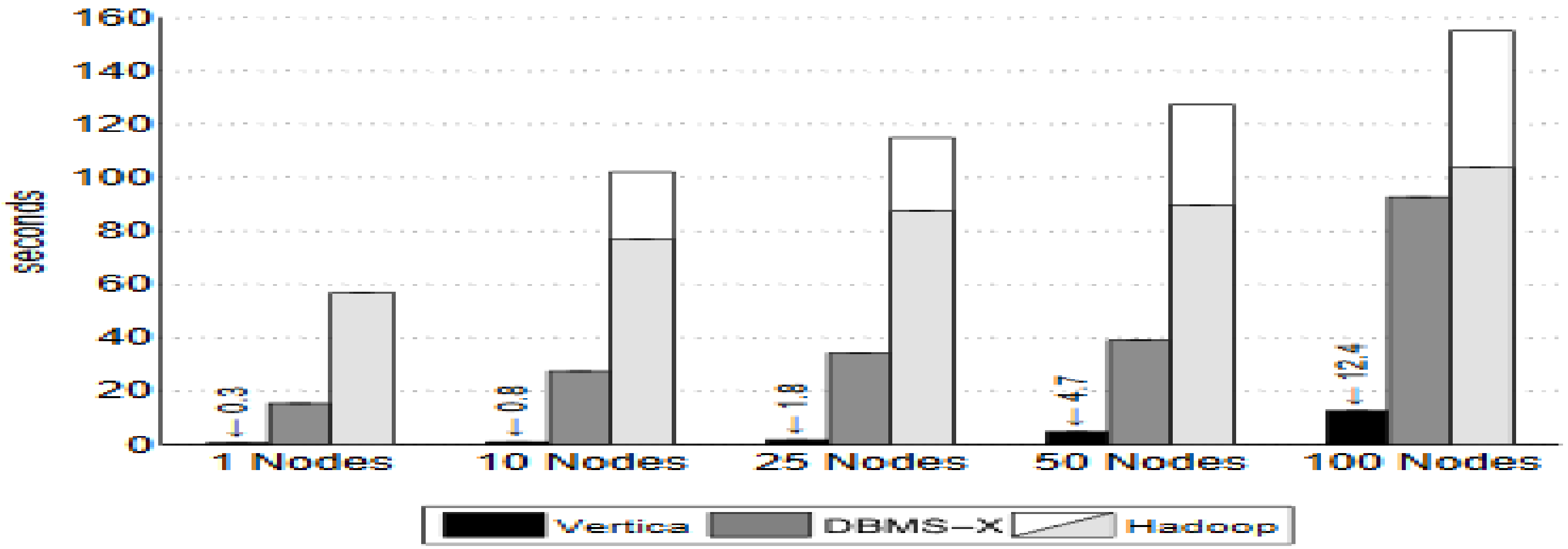


Figure 6: Selection Task Results

2- Analytical Tasks – Aggregation Task

- ✓ Designed to measure the performance of parallel analytics on a single read-only table
- ✓ **SQL** command: SELECT sourceIP, SUM(adRevenue) FROM UserVisits GROUP BY sourceIP;
- ✓ Also tested where SUBSTR(sourceIP, 1, 7) to measure the effect of having less groups
- ✓ Having less groups increases the speed as there are less number of groups for merging

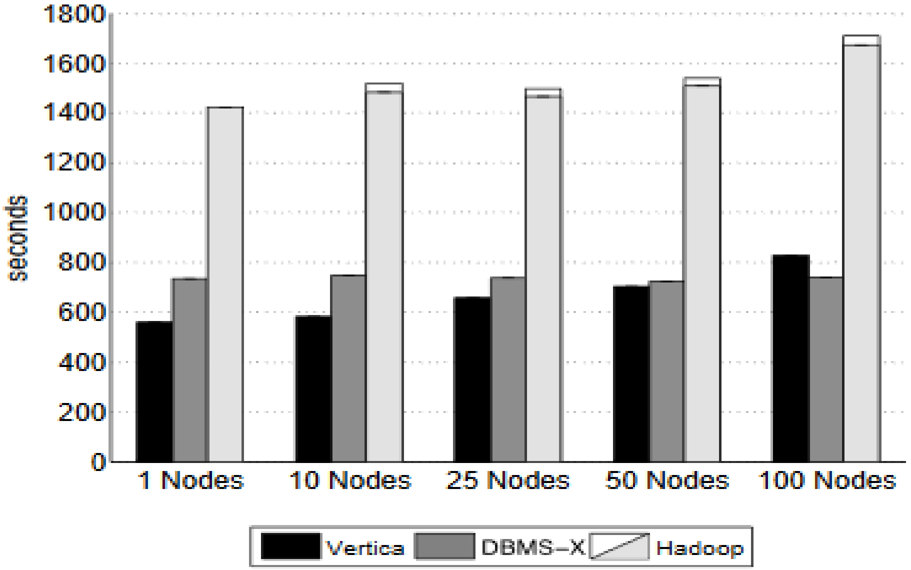


Figure 7: Aggregation Task Results (2.5 million Groups)

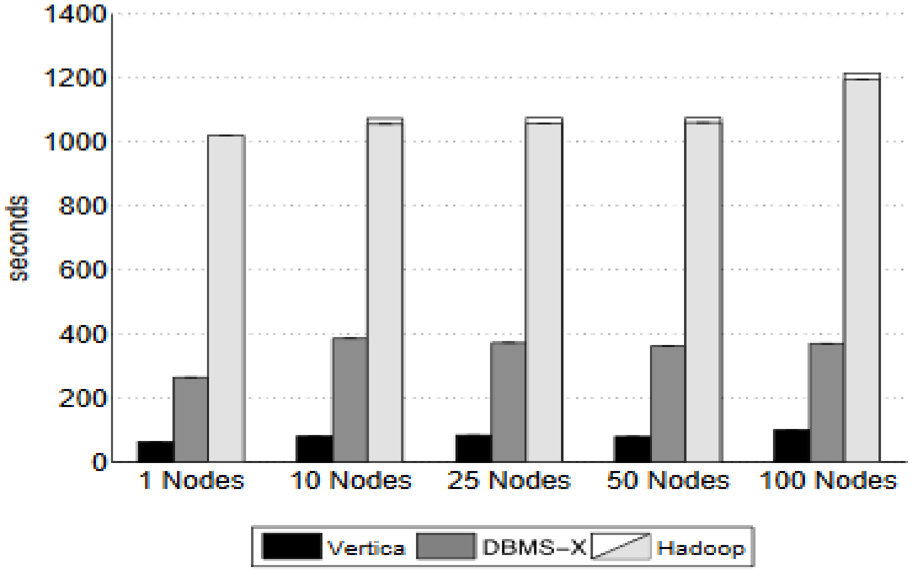


Figure 8: Aggregation Task Results (2,000 Groups)

2- Analytical Tasks- Join Task

- ✓ Find the performance of systems in dealing with large amount of calculation and joins
- ✓ Consists of two sub-tasks that perform complex calculation on two datasets

1. First find the source IP with the highest adrevenue among a particular date range
2. Then calculate the average Page Rank of all the pages that visited during this interval

2- Analytical Tasks- Join Task(Results)

- ✓ Parallel DBMS use clustered index on UserVisits.visitDate, Hadoop has to read the whole table from disk (20GB)
- ✓ Both tables in Parallel DBMSs made advantage of partitioning by Join Key.

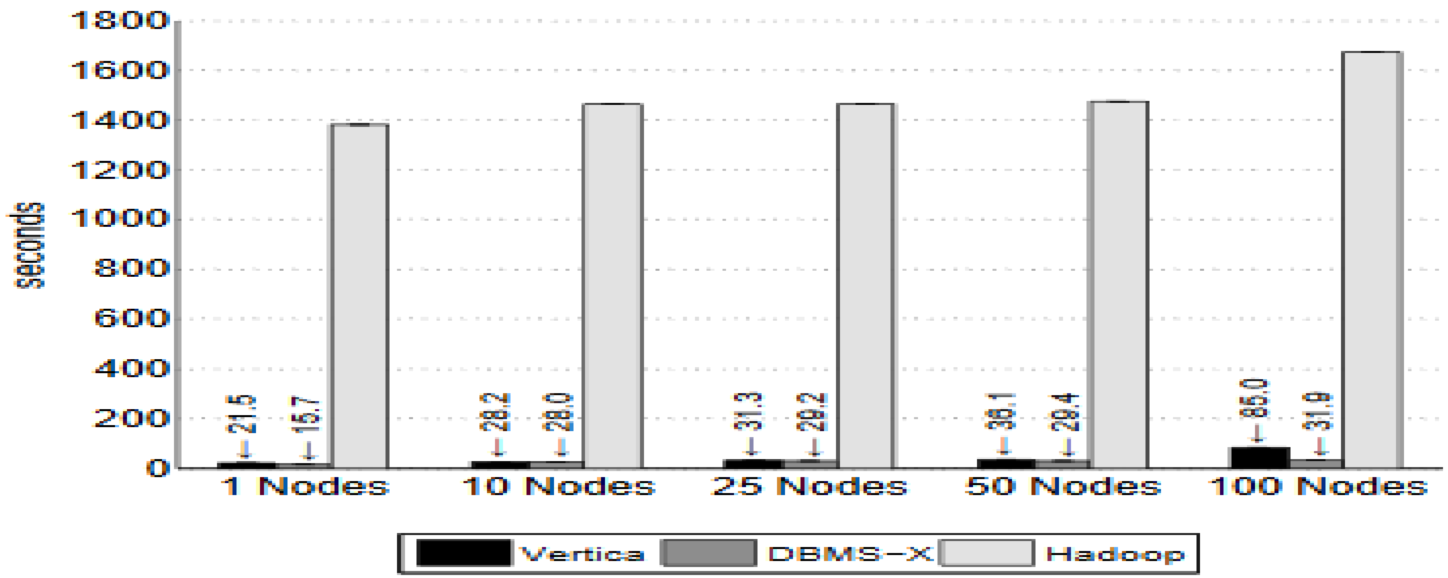


Figure 9: Join Task Results

2- Analytical Tasks- UDF Aggregation Task

- ✓ Search each document and find all the URLs that appear in the document
- ✓ for each unique URL, count the number of unique pages that reference that particular URL across the entire set of files
- ✓ Bottom part represents the UDF execution time, top bar represent the actual query time

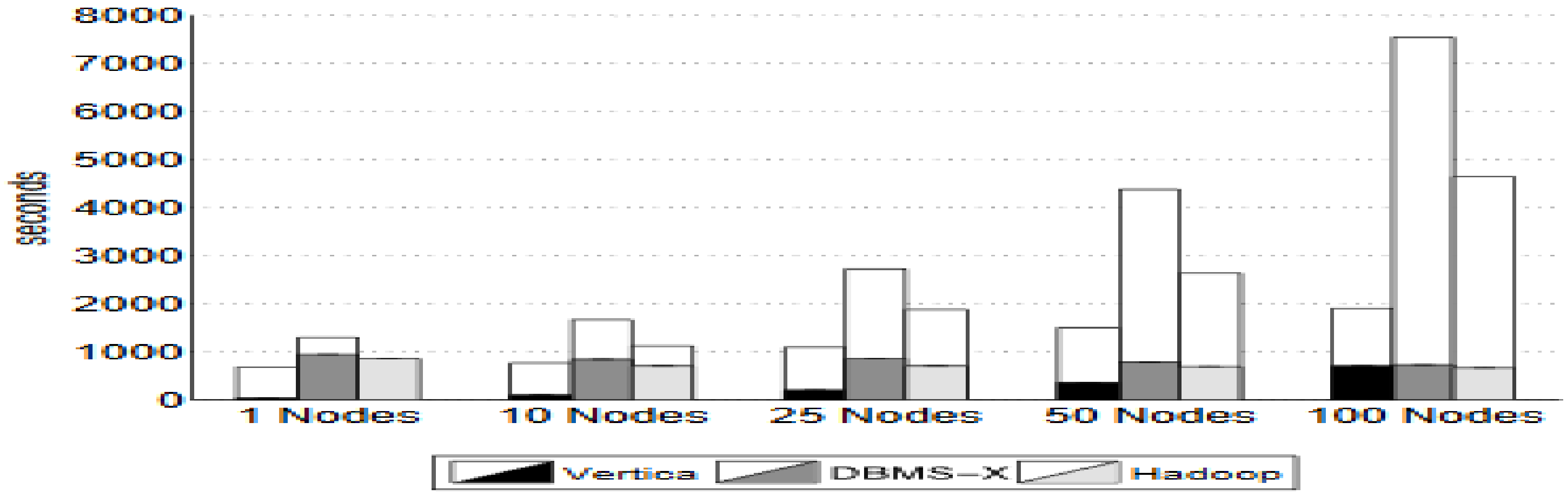


Figure 10: UDF Aggregation Task Results

What Happens in the system?

- ✓ System-level aspects
  - ✓ Installation, configuration, tuning
  - ✓ Task start-up
  - ✓ Compression
  - ✓ Loading data in and out
  - ✓ Execution strategies
  - ✓ Failure Model

- ✓ User-level aspects
  - ✓ Ease of use
  - ✓ Additional Tools

System Installation, Configuration, and Tuning

Hadoop installation & configuration is relatively easy. Only certain number of parameters influence the performance

DBMS-X has a GUI installation. Install on one node and use the generated file on all others

Vertica is installed by an RPM on each. Database tuning is minimal

- ✓ **Parallel Databases** are more difficult to setup and configure than **Hadoop**
- ✓ Changes in **Parallel databases** are done before the query execution
- ✓ In **Hadoop**, each individual task work best with a specific configuration.

Start-up and Compression

MR programs take sometimes to run at full-capacity.
parallel DBMSs are started at OS boot time and always ready to run the jobs. They can run multiple jobs at the same time.

Compression could result in a factor of 6–10 space savings in Parallel DBMSs.
If **executor** can operate directly on compressed data, compression is the obvious winner.
Hadoop's HDFS support both record-level and block-level compression.

Data Layout, Loading and Execution strategies

Parallel DBMSs have the opportunity to reorganize the input data file at load time.
MapReduce systems are unable to change the layout of data.

MR loading process is significantly faster. 3X faster than Vertica, 20X faster than DBMS-X

Query planner in parallel DBMSs are careful to transfer data between nodes only if it is absolutely necessary
MapReduce systems use a large number of control messages to synchronize processing

Failure Model

MapReduce is able to recover from faults in the middle of query execution in a way that most **parallel database** systems cannot.

Parallel Databases need to implement the same fault tolerant model.

User-level Aspects

- ✓ Both systems provide run-time support for debugging.
- ✓ Parallel DBMSs codes are in SQL, MapReduce are primarily in Java
- ✓ MR programming might be easier for developers
- ✓ In a long term Parallel DBMS are easier to deal with

- ✓ **MapReduce** programs have a web-interface to monitor the executions
- ✓ **Parallel DBMSs** have a lot of additional tools for **visualization**, **data mining,** etc.

How much does it cost?

	Vertica	HANA
Per 1TB, 256 GB	100,000$	250,000$

Cloudera with Hadoop is 7,000$ per node

Hadoop only will be about 4,000$

Conclusion

- ✓ Benchmarks showed Parallel DBMSs have a significant performance over Hadoop MapReduce.
- ✓ The advantages of Parallel Databases is the result of developed technologies in recent decades
- ✓ Hadoop **MapReduce** has an upfront cost advantage compared to the **Parallel** DBMSs
- ✓ New tools for Hadoop MapReduce such as **Hive** and **Spark** could be a game changer.

Thank You!

ANY QUESTIONS?