Introduction to Apache HIVE

Agenda

- **Background.**
- HIVE.
- HiveQL.
- Extension mechanisms.
- Performance comparison.

Motivation

- Analysis of Data made by both engineering and non-engineering people.
- The data are growing fast. In 2007, the volume was 15TB and it grew up to 200TB in 2010.
- Current RDBMS can NOT handle it.
- Current solution are not available, not scalable, Expensive and Proprietary.

Map/Reduce - Apache Hadoop

- MapReduce is a programing model and an associated implementation introduced by Goolge in 2004.
- Apache Hadoop is a software framework inspired by Google's MapReduce.

Motivation (cont.)

- Hadoop supports data-intensive distributed applications.

However...

  - Map-reduce hard to program (users know sql/bash/python).
  - No schema.

Agenda

- Background.
- **HIVE**
- HiveQL.
- Extension mechanisms.
- Performance comparison.

What is HIVE?

A data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis.

- ETL(Extract-Transform-Load).
- Structure.
- Access to different storage.
- Query execution via MapReduce.

Key Building Principles:

- SQL is a familiar language
- Extensibility – Types, Functions, Formats, Scripts
- Performance

Data Units

- Databases.
- Tables.
- Partitions.
- Buckets (or Clusters).

Type System

- Primitive types
  - Integers:TINYINT, SMALLINT, INT, BIGINT.
  - Boolean: BOOLEAN.
  - Floating point numbers: FLOAT, DOUBLE .
  - String: STRING.
- Complex types
  - Structs: {a INT; b INT}.
  - Maps: M['group'].
  - Arrays: ['a', 'b', 'c'], A[1] returns 'b'.

Agenda

- Background.
- HIVE.
- **HiveQL.**
- Extension mechanisms.
- Performance comparison.

Examples – DDL Operations

- **CREATE TABLE** sample (foo INT, bar STRING) **PARTITIONED BY** (ds STRING);
- **SHOW TABLES** '.*s';
- **DESCRIBE** sample;
- **ALTER TABLE** sample **ADD COLUMNS** (new_col INT);
- **DROP TABLE** sample;

Examples – DML Operations

- **LOAD DATA LOCAL INPATH** './sample.txt' **OVERWRITE INTO TABLE** sample **PARTITION** (ds='2012-02-24');

- **LOAD DATA INPATH** '/user/falvariz/hive/sample.txt' **OVERWRITE INTO TABLE** sample **PARTITION** (ds='2012-02-24');

SELECTS and FILTERS

- **SELECT** foo **FROM** sample **WHERE** ds='2012-02-24';
- **INSERT OVERWRITE DIRECTORY** '/tmp/hdfs_out' **SELECT** * **FROM** sample **WHERE** ds='2012-02-24';
- **INSERT OVERWRITE LOCAL DIRECTORY** '/tmp/hive-sample-out' **SELECT** * **FROM** sample;

Aggregations and Groups

- **SELECT MAX**(foo) **FROM** sample;
- **SELECT** ds, **COUNT**(*), **SUM**(foo) **FROM** sample **GROUP BY** ds;
- **FROM** sample s **INSERT OVERWRITE TABLE** bar **SELECT** s.bar, count(*) WHERE s.foo > 0 **GROUP BY** s.bar;

Join

```
CREATE TABLE customer (id INT,name STRING,address STRING)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY '#';
CREATE TABLE order_cust (id INT,cus_id INT,prod_id INT,price INT)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
```

- **SELECT** * **FROM** customer c **JOIN** order_cust o **ON** (c.id=o.cus_id);
- **SELECT** c.id,c.name,c.address,ce.exp **FROM** customer c **JOIN** **(SELECT** cus_id,sum(price) AS exp **FROM** order_cust **GROUP BY** cus_id**)** ce ON (c.id=ce.cus_id);

Agenda

- Background.
- HIVE
- HiveQL.
- Extension mechanisms.
- **Performance comparison.**

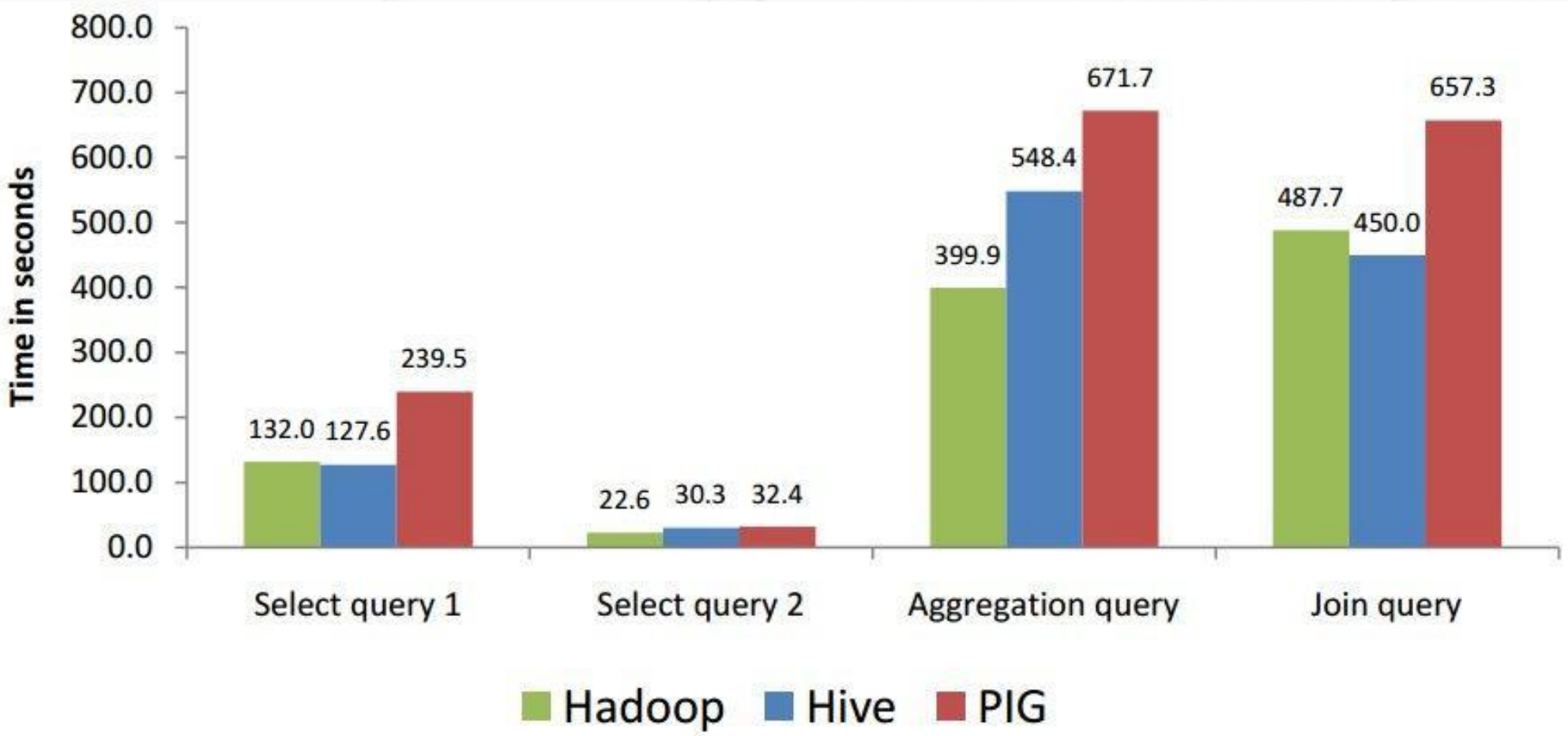
Performance - Dataset structure

grep(key VARCHAR(10), field VARCHAR(90))	2 columns, 500 million rows, 50GB
rankings(pageRank INT, pageURL VARCHAR(100), avgDuration INT)	3 columns, 56.3 million rows, 3.3GB.
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).	9 columns, 465 million rows, 60GB (scaled down from 200GB).

Performance - Test query

Select query 1	**SELECT** * **FROM** grep **WHERE** field like '%XYZ%';
Select query 2	**SELECT** pageRank, pageURL **FROM** rankings **WHERE** pageRank > 10;
Aggregation query	**SELECT** sourceIP, **SUM**(adRevenue) **FROM** uservisits **GROUP BY** sourceIP;
Join query	**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(`1999-01-01') AND Date(`2000-01-01') GROUP BY UV.sourceIP;

Performance - Result



Conclusion

- A easy way to process large scale data.
- Support SQL-based queries.
- Provide more user defined interfaces to extend Programmability.
- Files in HDFS are immutable. Tipically:
  - Log processing: Daily Report, User Activity Measurement
  - Data/Text mining: Machine learning (Training Data)
  - Business intelligence: Advertising Delivery,Spam Detection

HIGH PERFORMANCE SOLUTIONS

GLOBANT STUDIO

Thank you!

Globant

we are ready