HIVE

A warehouse solution
over
map-reduce framework

Dony Ang

Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao,
Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff and Raghotham Murthy

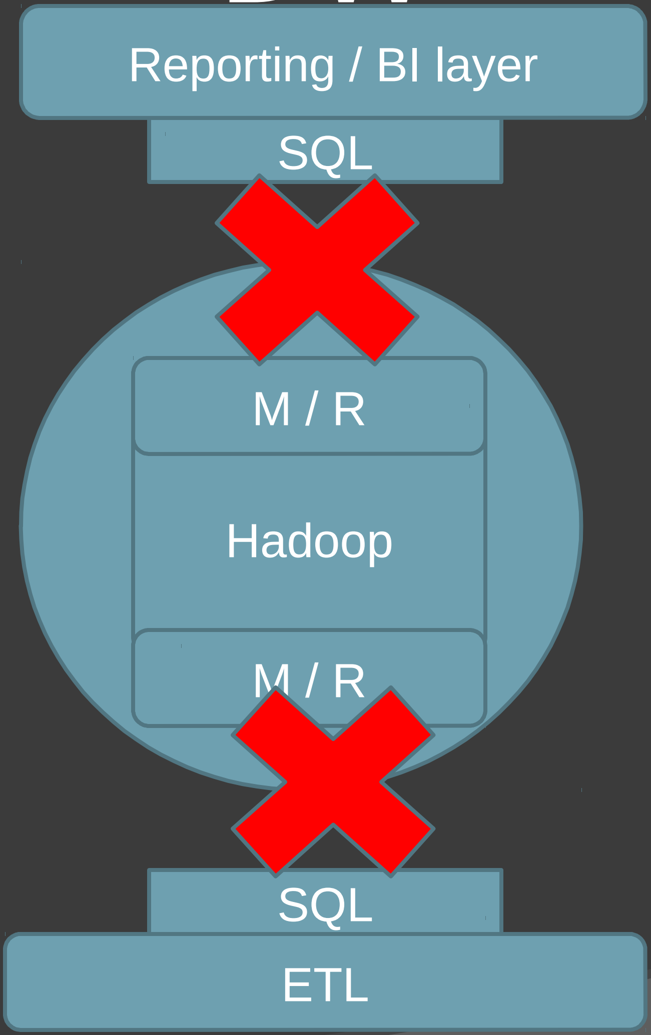
overview

- background
- what is Hive
- Hive DB
- Hive architecture
- Hive datatypes
- hiveQL
- hive components
- execution flows
- compiler in details
- pros and cons
- conclusion

background

- Size of collected and analyzed datasets for business intelligence is growing rapidly, making traditional warehousing more $$$
- Hadoop is a popular open source map-reduce as an alternative to store and process extremely large data sets on commodity hardware
- However, map reduce itself is very low-level and required developers to write custom code.

General Ecosystem of DW



what is hive ?

- Open-source DW solution built on top of Hadoop
- Support SQL-like declarative language called HiveQL which are compiled into map-reduce jobs executed on Hadoop
- Also support custom map-reduce script to be plugged into query.
- Includes a system catalog, Hive Metastore for query optimizations and data exploration

Hive Database

- Data Model
  - Tables
    - Analogous to tables in relational database
    - Each table has a corresponding HDFS dir
    - Data is serialized and stored in files within dir
    - Support external tables on data stored in HDFS, NFS or local directory.
  - Partitions
    - @table can have 1 or more partitions (1-level) which determine the distribution of data within subdirectories of table directory.

HIVE Database cont.

e.q : Table T under /wh/T and is partitioned on column ds + ctry

For ds=20090101

ctry=US

Then data is stored within dir /wh/T/ds=20090101/ctry=US

- Buckets
  - Data in each partition are divided into buckets based on hash of a column in the table. Each bucket is stored as a file in the partition directory.

HIVE datatype

- Support primitive column types
  - Integer
  - Floating point
  - Strings
  - Date
  - Boolean
- As well as nestable collections such as array or map
- User can also define their own type programmatically

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'.

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;

hiveQL

- Support SQL-like query language called HiveQL for select,join, aggregate, union all and sub-query in the from clause
- Support DDL stmt such as CREATE table with serialization format, partitioning and bucketing columns
- Command to load data from external sources and INSERT into HIVE tables.

```
LOAD DATA LOCAL INPATH ‘/logs/status_updates’
INTO TABLE status_updates PARTITION (ds=‘2009-03-20’)
```

- DO NOT support UPDATE and DELETE

hiveQL cont.

- Support multi-table INSERT

```
FROM (SELECT a.status, b.schoold, b.gender
        FROM status_updates a JOIN profiles b
          ON (a..userid = b.userid)
          and a.ds='2009-03-20')
     ) subq1
INSERT OVERWRITE TABLE gender_summary PARTITION (ds='2009-03-20')
  SELECT subq1.gender,COUNT(1) GROUP BY subq1.gender
INSERT OVERWRITE TABLE school_summary PARTITION (ds='009-03-20')
  SELECT subq.school, COUNT(1) GROUP BY subq1.school
```

- Also support User-defined column transformation (UDF) and aggregation (UDAF) function written in Java

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);

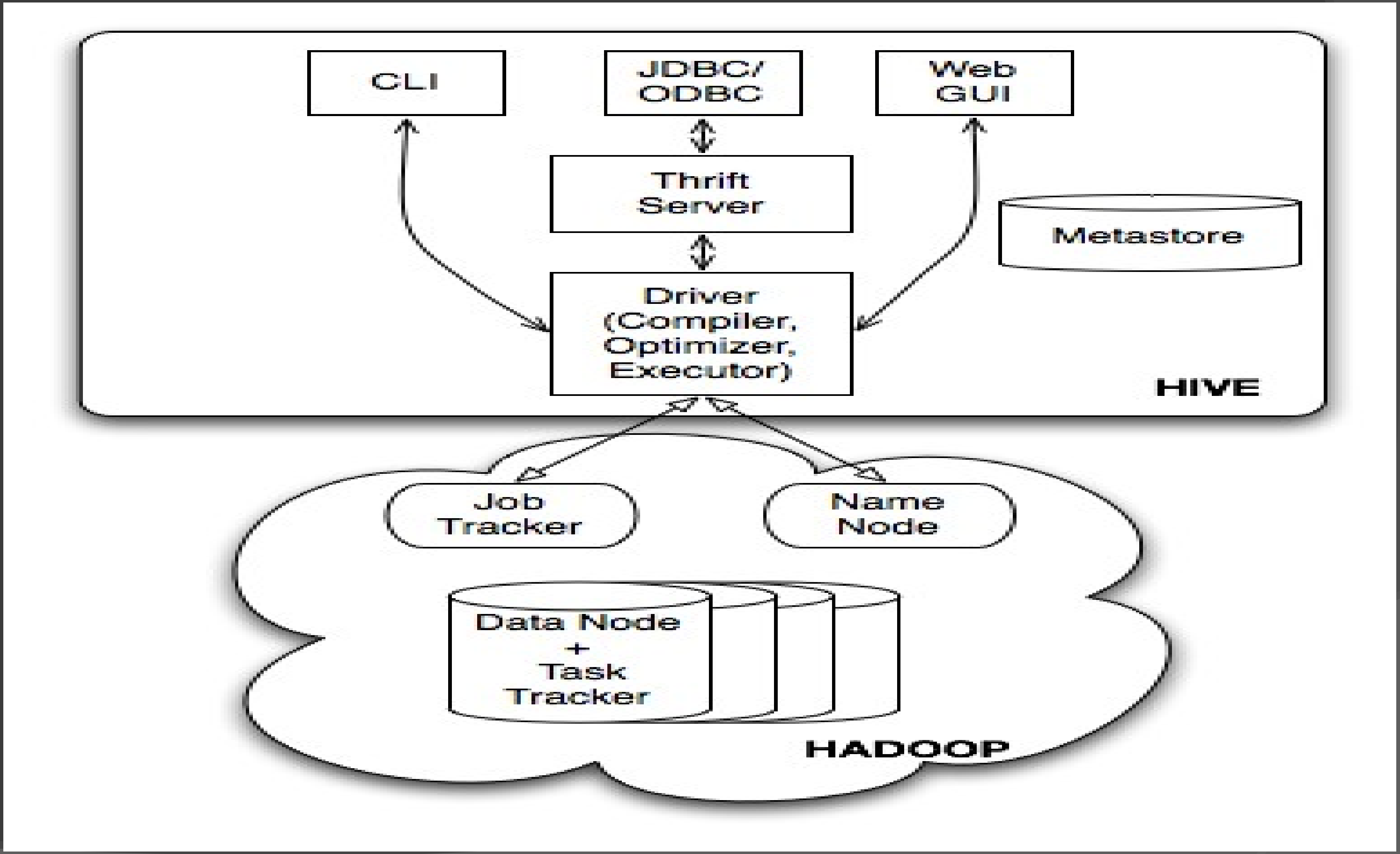
Multi table insert - Dynamic partition insert

```
 FROM page_view_stg pvs
   INSERT OVERWRITE TABLE page_view PARTITION(dt='2008-06-08', country='US')
          SELECT pvs.viewTime, … WHERE pvs.country = 'US'
   INSERT OVERWRITE TABLE page_view PARTITION(dt='2008-06-08', country='CA')
          SELECT pvs.viewTime, ... WHERE pvs.country = 'CA'
   INSERT OVERWRITE TABLE page_view PARTITION(dt='2008-06-08', country='UK')
          SELECT pvs.viewTime, ... WHERE pvs.country = 'UK';

FROM page_view_stg pvs
   INSERT OVERWRITE TABLE page_view PARTITION(dt='2008-06-08', country)
          SELECT pvs.viewTime, ...
```

https://cwiki.apache.org/confluence/display/Hive/Tutorial#Tutorial-Dynamic-PartitionInsert

HIVE Architecture



HIVE Components

- External Interfaces
  - User Interfaces both CLI and Web UI and API likes JDBC and ODBC.
- Hive Thrift Server
  - simple client API to execute HiveQL statements
- Metastore – system catalog
- Driver
  - Manages the lifecycle of HiveQL for compilation, optimization and execution.

Execution Flow