B0M33BDT
Big Data technologies

Map Reduce and Hive

Sergii Stamenov, Marek Sušický

6. October 2021

Outline

PROFINIT

1. How to store data on Hadoop and storage formats
2. How to process data on Hadoop
   1. MapReduce
   2. Hive

Storage

Data access patterns

PROFINIT

- OLTP
  - Online transaction processing is the heart of any IT application
  - Touches only a single (a very few) rows at the time
  - Typical operations: insert, update, delete
  - “As a user I want to place a new order with X, Y, Z items”

- OLAP
  - Online analytical processing
  - Allows business to answer a question “what is going on”
  - Touches large fraction of rows in a table (all or a specific subset/segment)
  - Typical operations: read
  - “What is the average sum a customer pays us monthly”

Data exchange formats

PROFINIT

- Plain text
  - XML
  - json
  - separated (CSV, TSV) etc.

- Binary
  - Various proprietary formats (Excel, pdf, protobuf)
  - Multimedia files (image, video, sound)

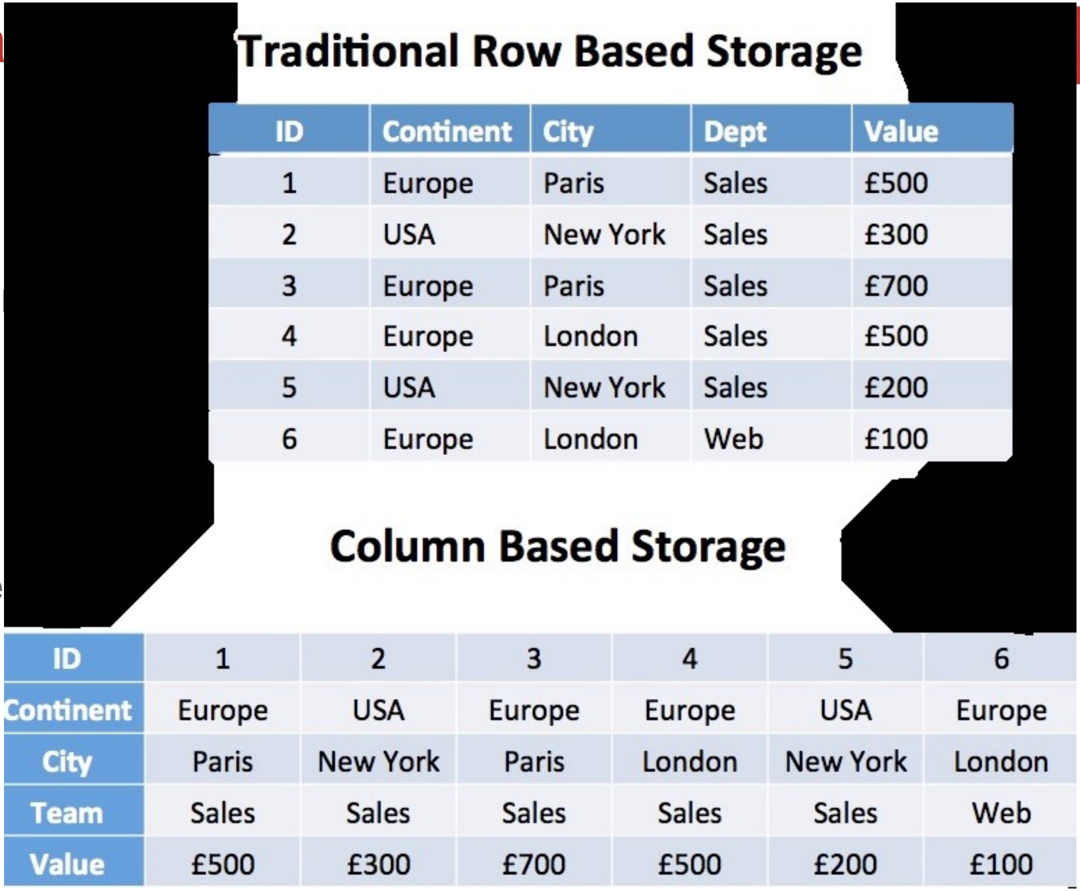
Data exchange formats considerations

PROFINIT

- Plain text
  - Self-contained and human-readable, no need for special software to understand/change the content ✓
  - Takes more space on disk compared to binary formats ✕
- Binary
  - Efficient use of disk space (data types and compression) ✓
  - Use of data types allows data validation and data integrity ✓
  - Is a must when performance is key ✓
  - Requires specialized tool to read/change ✕

Tabular data forma

- Row-oriented
  - Faster writes
  - Works best for OLTP
- Columnar
  - Optimal reads
  - More efficient compre
  - OLAP



Traditional Row Based Storage

ID	Continent	City	Dept	Value
1	Europe	Paris	Sales	£500
2	USA	New York	Sales	£300
3	Europe	Paris	Sales	£700
4	Europe	London	Sales	£500
5	USA	New York	Sales	£200
6	Europe	London	Web	£100

Column Based Storage

ID	1	2	3	4	5	6
Continent	Europe	USA	Europe	Europe	USA	Europe
City	Paris	New York	Paris	London	New York	London
Team	Sales	Sales	Sales	Sales	Sales	Web
Value	£500	£300	£700	£500	£200	£100

Hadoop data formats

- AVRO
  - Binary row format (+ json schema definition)
  - Has schema and schema evolution support
  - Kafka
- ORC
  - Binary columnar format
  - Well integrated into Hive (optimizations, data types, compression)
- **Parquet**
  - Binary columnar format
  - Schema is a part of file, nested objects are also supported
  - Has wide adoption and good performance on different workloads

Compression

- Motivating example:
  - How long does it take to read 100GB table in 10 nodes cluster (each node has 4 disk with peak read speed 100MB/s)?
- Helps us to trade I/O time for CPU time
- Practical considerations is tradeoff between compression coefficient and compression/decompression speed
  - Gzip – very efficient in terms of compression, but is relatively slow
  - Snappy – good balance between compression efficiency and speed

MapReduce

MapReduce

- Paradigm / framework for distributed computation
- Consists of 2 phases/functions:
  - `map(key: object, value: object) -> Tuple[object, object]`
  - `reduce(key: object, values: List[object]) -> Tuple[object, object]`
- Example: Lets count the number of students signed up for a course

student	course
Alice	BDT
Bob	BDT
Cate	ABC

→ map

course	value
BDT	1
BDT	1
ABC	1

→ shuffle

course	values
BDT	[1, 1]
ABC	1

→ reduce

course	value
BDT	2
ABC	1

Inputs

PROFINIT

- Usually an input for a MapReduce job is a directory on HDFS
- The input is divided into data blocks of fixed sized called input splits
- For each split a map task is created
- Map tasks can run on the same or different machines allowing us to scale data pipeline as needed
- Hadoop tries to allocate a map task next to a data block of the input if possible (data locality principle)

Mapper

- The mapper function is called once for the input row and it can generate any number of output key-value pairs (even none)
- The mapper function is stateless
- The intermediate results are sorted by key and written to local disk
- We can apply reduce operation on map-side to reduce amount of data transferred over the network, this operation is called **Combiner**

Shuffle

- There might be more than one map task that processed data with a specific key
- It is responsibility of the shuffle stage to make sure that all map-outputs with a specific key are delivered to a single reducer
- Each result file is partitioned and sorted before it is sent to a reducer

Reduce

- Transforms the output of shuffle stage to final result
- Reduce task download partial results files from mappers and merge-sort them before processing
- Number of reduce tasks is determined by the job author
- We can set the number of reduce tasks to 0

Parallelization

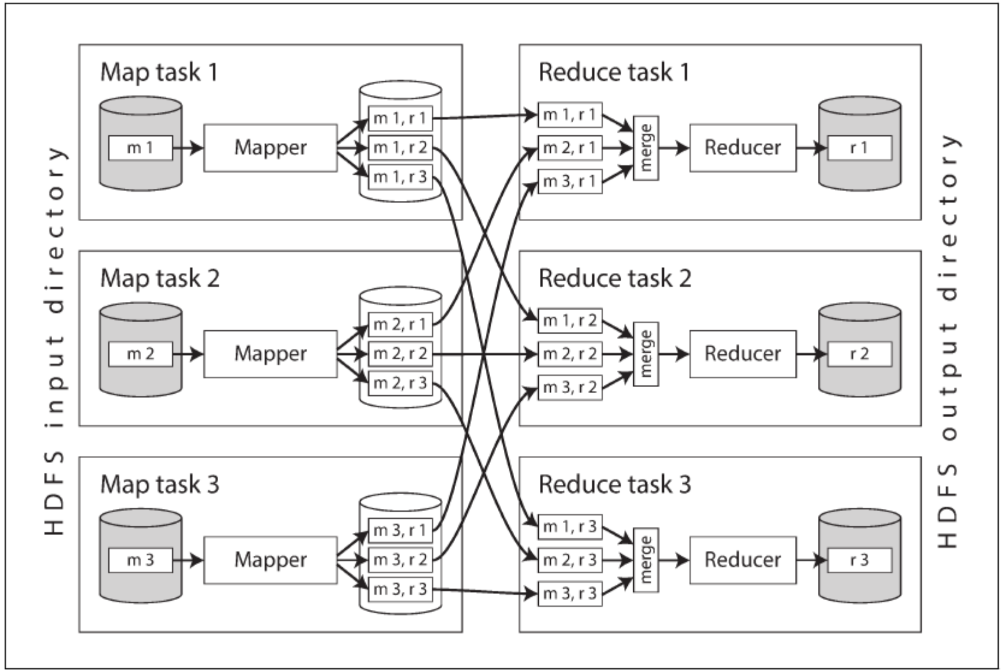
PROFINIT

- Reduce operations should be
  - Associative (A x B) x C = A x (B x C) (since values for the same key aren’t sorted)
  - Neutral “zero” element should exist
  - Optionally: Commutative (A x B) = (B x A) (allows combine operations)

- Typical operations include
  - min, max
  - count, sum, multiplication
  - string concatenation
  - set union and intersection

MapReduce task anatomy

PROFINIT



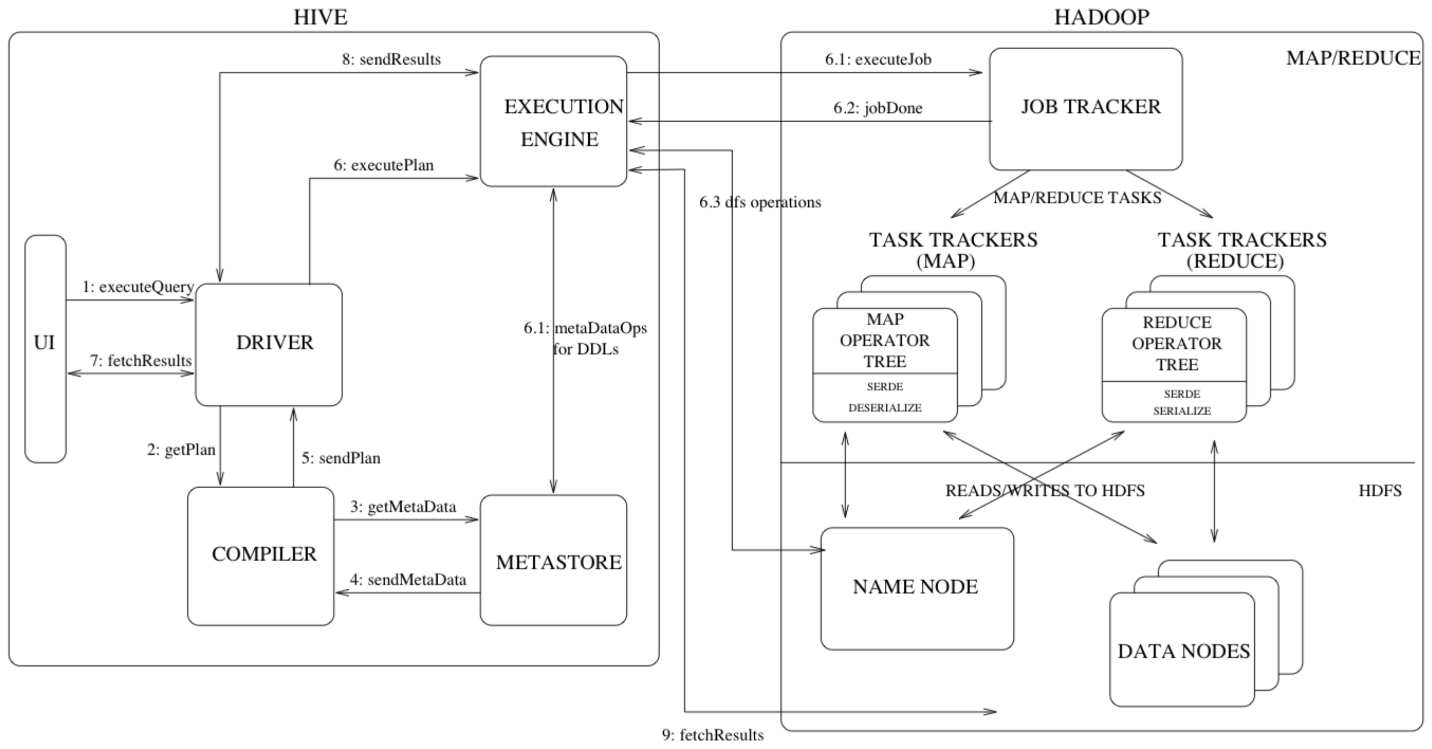
Hive

Hive

- MapReduce is a big step towards easier distributed computation, but requires a lot of coding in Java even for simple counting
- SQL is *lingua franca* for data analytics
- Apache Hive is a SQL-engine built on top of MapReduce

High-level architecture

PROFINIT



Hive - data

- Data is organized into tables stored on HDFS
  - Table's data files are stored in a HDFS directory
  - Schema on read – schema is checked during the query
- A table is metadata stored in the metastore. Metastore contains:
  - Table schema
  - Table data location and format
  - Custom attributes
  - statistics

Hive compared to relational DBs

- Schema-on-read
- Indexing is not supported*
- Limited support for transactions and isolation
- Materialized views are not supported

**initial design had flaws, usage was discouraged and indexing was removed in Hive 3.0*

Hive - HQL

PROFINIT

- DDL (Data Definition Language)
  - `CREATE [EXTERNAL] TABLE`
  - `DROP TABLE, TRUNCATE TABLE, ALTER TABLE`
- DML (Data Manipulation Language)
  - `LOAD DATA, INSERT INTO TABLE, INSERT OVERWRITE TABLE`
- Query
  - `SELECT`
- Limited support*
  - `UPDATE`
  - `DELETE`

**supported only in transactional tables*

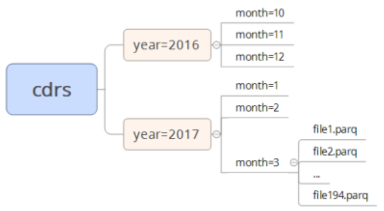
Hive – loading data

PROFINIT

- Hive defines multiple ways to load data
- Load data from HDFS using LOAD DATA statement
  - HDFS cp/mv operations, schema is not checked during load
- Insert query results into a table using INSERT INTO table select * from tbl
- Inserting literal values using INSERT INTO table values (1, 2, 3)
  - Least efficient way to insert values into a Hive, use this only for testing
  - Every insert statement will create a single (small) file

Partitioning & Bucketing

- Partitioning – a way to organize data into smaller chunks
  - Logical and physical separation
  - Can speed-up some queries
  - Simplify governance



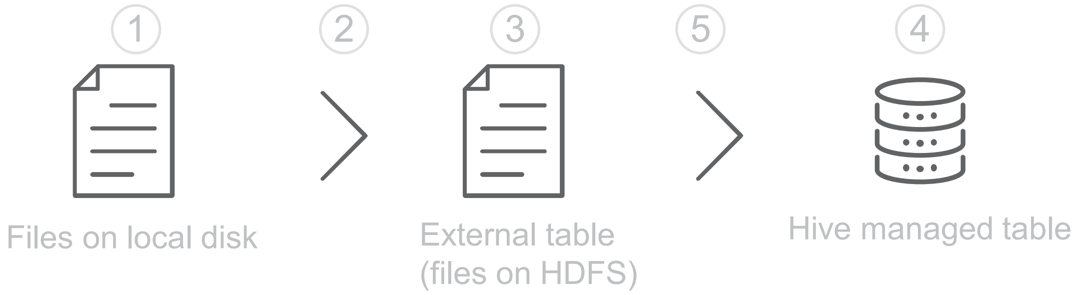
- Design partitioning schema with the data volume in mind, we do not want to have too many small files
  - If we don't have enough data, daily partitioning might not be very efficient
- Bucketing – additional layer of organization data into files by using hash function applied on bucketed column.
  - We can make sure that rows with the same bucketing key will be in the same file

Execution engines

- MapReduce
- Hive on Tez
  - Optimized query execution that avoid some limitations of MapReduce
  - Eliminate unnecessary stages and HDFS writes
- Hive on Spark
  - Another approach onto speeding up MapReduce jobs by translating it into Spark jobs
- LLAP
  - Live Long And Process daemons for small/short queries

Hive example workflow

PROFINIT



1. Raw data is delivered to front-end server
2. Raw data is copied to HDFS folder for raw data
3. An external table is registered in Hive
4. An internal table is created in Hive (optimized file format)
5. Data is transformed and inserted into internal table

External table

PROFINIT

```
CREATE EXTERNAL TABLE IF NOT EXISTS ap_temp (
   ACC_KEY BIGINT,
   BUS_PROD_TP_ID VARCHAR(255),
   START_DATE TIMESTAMP,
   BUS_PROD_TP_DESCR VARCHAR(255)
)
ROW FORMAT
   DELIMITED FIELDS TERMINATED BY '~'
   LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION '/data/input/acc';
```

Internal table

```
CREATE TABLE IF NOT EXISTS ap (
   ACC_KEY BIGINT,
   BUS_PROD_TP_ID VARCHAR(255),
   START_DATE TIMESTAMP)
PARTITIONED BY (BUS_PROD_TP_DESCR VARCHAR(255))
CLUSTERED BY (ACC_KEY) INTO 32 BUCKETS
STORED AS ORC tblproperties ("orc.compress"="ZLIB");
```

Insert data

```
INSERT OVERWRITE TABLE ap
PARTITION (BUS_PROD_TP_DESCR)
SELECT
 ACC_KEY,
 BUS_PROD_TP_ID,
 START_DATE,
 BUS_PROD_TP_DESCR
FROM ap_temp;



DROP TABLE ap_temp;
```

Hive catalog

PROFINIT

- Hive defines statements that return information about databases and tables
  - show databases
  - show tables <db_name>
  - show create table <table>
  - show partitions <table>
  - describe <table>
  - show columns from <table>

Summary

- Use columnar formats for analytics workloads
- Use row-wise format when you anticipate full-scans or you need to access whole row at once
- Try to make files in fit a HDFS block-size
- Use partitioning

Thank you

PROFINIT

Profinit EU, s.r.o., Tychonova 2, 160 00 Praha 6
Tel.: + 420 224 316 016, web: www.profinit.eu

LinkedIn
linkedin.com/company/profinit

Twitter
twitter.com/Profinit_EU

Facebook
facebook.com/Profinit.EU

Youtube
Profinit EU