Lecture 30:

• SQL on MapReduce Example: Apache Hive

Announcements:

• R-4 out (due Fri)
• HW-5 out
• Quiz 5: Traditional SQL engines (trees, joins, costs), Data Formats (layouts, details)
• Exam 2: Next Friday (4/19)
• Last Reading: Out Friday-ish
• Last Assignment: Out next Wednesday-ish

Hive and Presto

Quick Notes on Facebook/Meta:

• Like major cloud providers, build & run their own data centers
• Currently have 23+ across US, Asia, and Europe
• First was in Prineville, OR (circa 2010)
• Have developed various systems for cluster computing ...

Embraced open-source throughout their existence

• Developed and contributed to various projects
• Some big data / database related projects: Hive, Presto, RocksDB, Velox
• ML / AI related: PyTorch, LLAMA
• Plus many more
Apache Hive Intro

Hive Basics: ... from: Thusoo et al., 2009

- Initially used at FB as a data warehouse (in paper, over 700TBs)
- As FB adopted Hadoop, still wanted ease of SQL (vs complex MR scripts)
- Hive compiles SQL queries to Hadoop MapReduce jobs ... HiveQL
- Also includes a system catalog (metadata store) ... Hive-Metastore

We’ll go over the basics ... as an example of SQL → MR approaches

Hive Data

Tables: Each table has a corresponding HDFS directory
- data serialized and stored in files (in directory/subdirectory)
- support different and custom serialization formats (record in catalog)

Partitions: Each table can have 1 or more partitions
- represented as subdirectories
- e.g.: my_table/year=2009/state=WA/ for year, state partitioning

Buckets: Partition data can be further divided
- each bucket is a hash of a column
- stores as subdirectory file
**Hive Architecture Overview**

**Metastore**: acts as the system catalog
- maintains namespaces of tables
- table columns and types, owner, and serialization info
- additional user-supplied (key-value) metadata
- location of table data in HDFS, data formats, bucket info

**Driver**: manages life cycle of HiveQL statements
- receives a statement (various interfaces)
- invokes compiler and submits for execution
- creates session handles to track statistics
- ... e.g., query execution time, number of output rows, etc.
Compiler: translates HiveQL statements into plans
- a plan is a directed acyclic graph (DAG) of map-reduce jobs
- has a parser, semantic analyzer, logical/physical plan generator, optimizer
- ... more soon

Execution Engine: HiveQL statements executed in Hadoop
- the driver supplies the optimized physical plan
- which is run as a sequence of map-reduce jobs within Hadoop
**Parser**: takes query string and outputs a parse tree
- HiveSQL is “SQL-like” with additional constructs
- Select-project-join, aggregates, group by, union, sort, subqueries (from)
- Constructs for creating partitions, buckets, data loading, etc
- Supports user-defined functions and map-reduce scripts

**Semantic Analyzer**: transforms parse tree to internal representation
- Uses schema information in Metastore for tables
- ... to “type check” query
- ... to expand `SELECT *`
- ... to automatically coerce data types
**Logical Plan Generator**: transforms IR to logical plan

- as a tree of logical operators