Hive

Cliff Engle
Cloud Computing, Fall 2011

Motivation for Hive:

- TB's of incoming raw data per day
  - Data is structured, but no metadata explicitly specified
- Scalability of Hadoop over traditional ACID databases
  - Running jobs on thousand node clusters is impractical for ACID db's
- It is difficult and tedious to manually write Map-Reduce code
  - A simple SQL query could require hundreds of lines of code in Map-Reduce

Hive's Goals

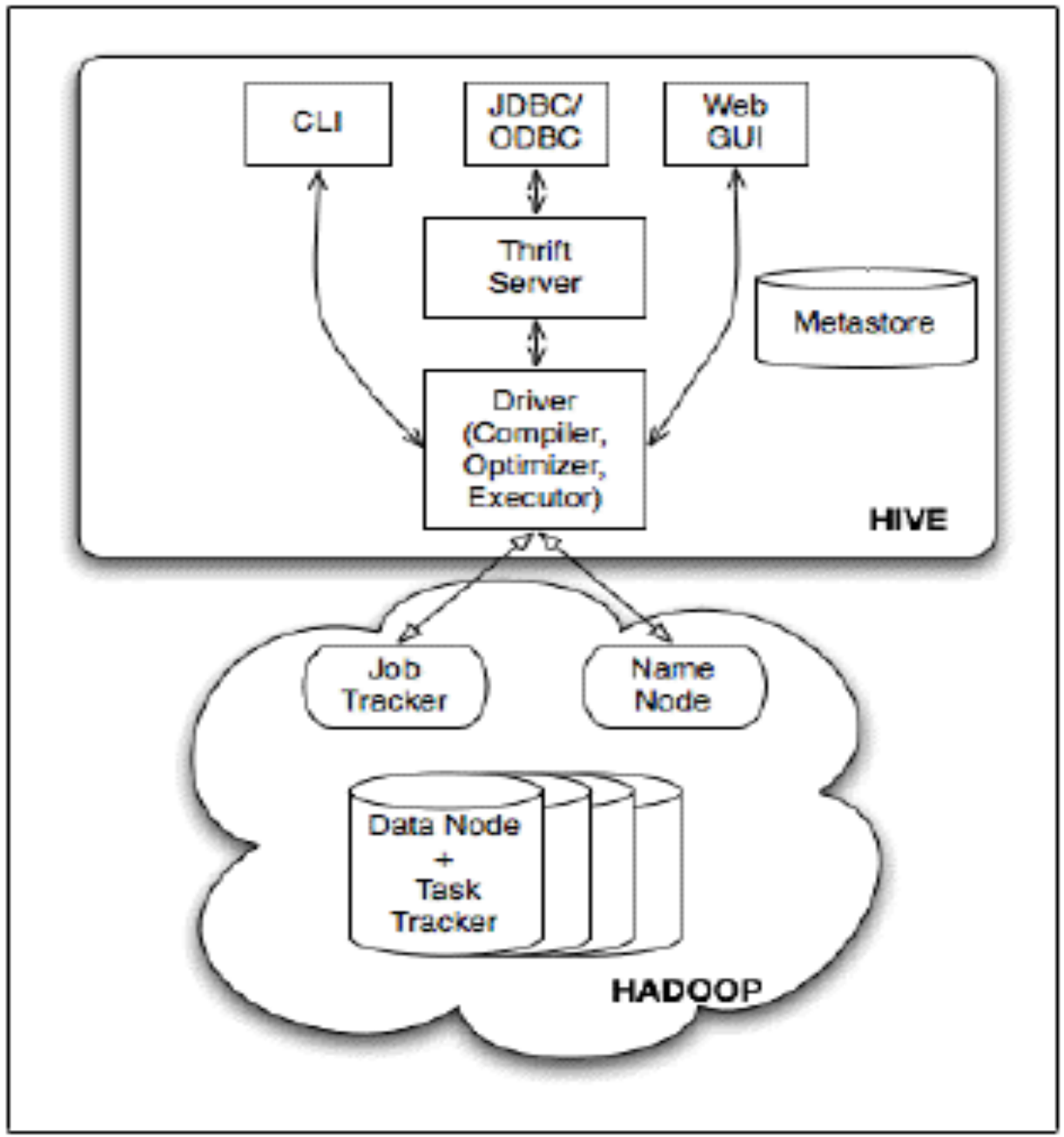
1. Associate a schema with data in HDFS
2. Provide a higher level language, HiveQL, to query the data
3. Flexibility with existing data
    1. User-Defined Functions (UDF's)
    2. Custom Serializer/Deserializer (SerDe's)
    3. Custom types
    4. Multiple storage formats (text, binary)
    5. Nested data (lists, maps, etc.)

What is Hive?

A system for managing and querying structured data built on top of Hadoop.

- HDFS for storage
- Map-Reduce for execution
- RDBMS for Metadata

Hive Architecture



Hive Architecture

- Multiple ways to initiate queries
  - CLI
  - JDBC
  - Thrift
- Queries parsed, optimized, compiled to M/R tasks, and executed on existing M/R cluster
- Metadata stores database names, table names, partitions

Compiler

Parser: HiveQL -> Abstract Syntax Tree (AST)

Semantic Analyzer: AST -> Logical Operator Tree

- Consists of operators like TableScan, Filter, Join, GroupBy

Logical Optimizer: Rewrite operator tree to more optimized one

- Predicate pushdown: Move filter closer to data source

Physical Plan Generator: Logical Op Tree -> MapReduce Tasks

Physical Optimizer: Support for Map-Side Join

- If one table in a join is small, it can be broadcast to all mappers for local joins instead of requiring a shuffle.

Additional Features

Indexing

- Compact Index
  - Stores block-offsets of indexed rows in a hidden Metadata table
  - Rewrites queries to use indexed table rather than perform full table scan
- Bitmap
  - Speeds up queries that are filtered on multiple columns

Partial aggregation for Group By

- Mappers perform hash-based aggregation before shuffle

Partitioning data

- Less data needs to be loaded from HDFS
- Sub-directory of main HDFS folder

Discussion

What does Hive do well?

- Open and extensible system
- Easy "out of the box" functionality on top of existing Hadoop/HDFS cluster.
  - vs. RDBMS that require all data to be loaded into DB, along with DB specific tuning knowledge
- Scalability
- Some optimizations, though fairly basic
  - Only rule based. No cost based optimizations

Does Hive fulfill its use cases?

- Automatically generated reports?
  - Yes, Hive is pretty good for daily or weekly reports since it runs batch Map-Reduce jobs.

- Ad hoc querying?
  - Not particularly. Query results are not real-time, so do not allow extensive exploratory queries. Queries are on the order of minutes rather than seconds.

Discussion

Hive jobs are not realtime. MapReduce takes ~30 seconds for null job.

- Google has Dremel for low-latency read-only queries

Is MapReduce a good fit for read-only queries or is a more flexible model better?

- DryadLINQ allows for more flexibility in joins/group-by's and additional optimizations

Should data warehouse manage data storage format as in traditional RDBMS or simply build on top of existing storage?

SQL vs domain specific language (Pig)

- Decouple language from data

Future Impact

- High level languages for querying read-only data on distributed systems have become quite popular. MS Scope and Yahoo Pig provide very similar functionality
- Stonebraker believes Map-Reduce is a step backwards
  - Hive offers some optimizations based on table schema, though not as significant as RDBMS
  - Stonebraker says 100 nodes is sufficient for 1-2PB range with traditional DBMS
  - Complaints about lack of indexes in M/R => Hive now supports simple indexes