SURVEY PAPER ON HIVE BIGDATA

Mukta Pujar

M.Tech II SEM (CSE)

Department of Studies in Computer Science& Engineering

The University B.D.T. College of Engineering,Davanagere – 577004, Karnataka, India

(A Constituent College of Visvesvaraya Technological University,Belagavi)

Mohammed Rafi

Department of Studies in Computer Science& Engineering

The University B.D.T. College of Engineering, Davanagere – 577004, Karnataka, India

(A Constituent College of Visvesvaraya Technological University, Belagavi)

Abstract

The size of data sets being collected and analyzedin the industry for business intelligence isgrowing rapidly, making traditional warehousing solutions prohibitively expensive. Hadoop is a popular open-source map-reduce implementation which is being used in companies like Yahoo, Facebook etc. to store and process extremely large data sets on commodity hardware. However, the map-reduce programming model is very low level and requires developers to write custom programs which are hard to maintain and reuse. In this paper, we present Hive, an open-source data warehousing solution built on top of Hadoop. Hive architecture, working, advantages and disadvantages. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Keywords-Hive, HiveQL, Hadoop, map-reduce, HDFS, HBASE.

Introduction

Big data is immensely growing with the increase in users on various social networking sites or in industries. Big Data can be classified in the form of structured, unstructured and semistructured form. Traditional data processing techniques are unable to store and process this huge amount of data, and due to this, they face many challenges in processing Big Data and demands of the end user. Nowadays web traffic, social media content, system data and machinegenerated data are growing rapidly.

Big data analytics helps organizations harness their data and use it to identify new opportunities. That, in turn, leads to smarter business moves, more efficient operations, higher profits and happier customers. The volume of data with the speed it is generated makes it difficult for the current computing infrastructure to handle big data.

Prerequisites:

Before proceeding with Hive, you need a basic knowledge of Core Java, Database concepts of SQL, Hadoop File system, and any of Linux operating system flavours.

Hadoop:

Hadoop is an open-source framework to store and process Big Data in a distributed environment. It contains two modules, one is MapReduce and another is Hadoop Distributed File System (HDFS).

i.MapReduce: It is a parallel programming model for processing large amounts of structured, semi-structured, and unstructured data on large clusters of commodity hardware.

ii. HDFS: Hadoop Distributed File System is a part of Hadoop framework, used to store and process the datasets. It provides a fault-tolerant file system to run on commodity hardware

The Hadoop ecosystem contains different sub-projects (tools) such as Sqoop, Pig, and Hive that are used to help Hadoop modules.

i. **Sqoop:** It is used to import and export data to and from between HDFS and RDBMS
ii. **Pig:** It is a procedural language platform used to develop a script for MapReduce operations.
iii. **Hive**: It is a platform used to develop SQL type scripts to do MapReduce operations.

What is Hive?

- Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.
- Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

Hive is not:

- A relational database
- A design for On-line Transaction Processing (OLTP)
- A language for real-time queries and row-level updates

DATA TYPES

Similar to traditional databases, Hive stores data in tables, where each table consists of a number of rows, and each row consists of a specified number of columns. Each
column has an associated type. The type is either a primitive type or a complex type. Currently, the following primitive types are Supported.

- Integers – bigint(8 bytes), int(4 bytes), smallint(2 bytes), tinyint(1 byte). All integer types are signed.
- Floating point numbers – float(single precision), double(double precision)
- String

Hive also natively supports the following complex types:

- Associative arrays – map<key-type, value-type>
- Lists – list<element-type>
- Structs – struct<file-name: field-type, ... >

Hive is an open-source data warehousing solution built on top of Hadoop. Hive's biggest advantage is that it supports queries expressed in a SQL-likedeclarativelanguage-HiveSQL,whichare easily compiled into map-reducejobsthatareefficientlyexecuted using Hadoop. In addition, custom map-reduce scripts can be plugged into queries, supported by HiveQL. Hive also includesa systemcatalog, Hive-Metastore, containing schemas and statistics, which is useful in data exploration and query optimization

Background

The entire data processing infrastructure in Facebook prior to 2008 was built around a data warehouse built using a commercial RDBMS. The data that were being generated was growing very fast -as an example from a 15TB data set in 2007 to a 700TB data set. The infrastructure at that time was so inadequate that some daily data processing jobs were taking more than a day to process and the situation was just getting worse with every passing day. Thus, people had an urgent need for infrastructure that could scale along with their data. As a result the Facebook team started exploring Hadoop as a technology to address their scaling needs. The fact that Hadoop was already an open source project that was being used at petabyte scale and provided scalability using commodity hardware was a very compelling proposition for them. The same jobs that had taken more than a day to complete could now be completed within a few hours using Hadoop. However, using Hadoop was not easy for end users, especially for those users who were not familiar with map-reduce. End users had to write map-reduce programs for simple tasks like getting raw counts or averages. Hadoop lacked the expressiveness of popular query languages like SQL and as are resultusersendedupspending hours (if not days) to write programs for even simple analysis.It was very clear to us that in order toreallyempowerthecompany to analyze thisdata more productively,peoplehadtoimprovethe query capabilities of Hadoop. Bringing this dataclosertousersis what inspired them to build Hive in January 2007. Their vision was to bring the familiar concepts of tables, columns, partitions and a subset of SQL to the unstructured world of Hadoop, while still maintaining the extensibility and flexibility that Hadoop enjoyed. Hive was open sourced in August 2008 and since then has been used andexploredbyanumberofHadoopusersfortheir data processing needs.

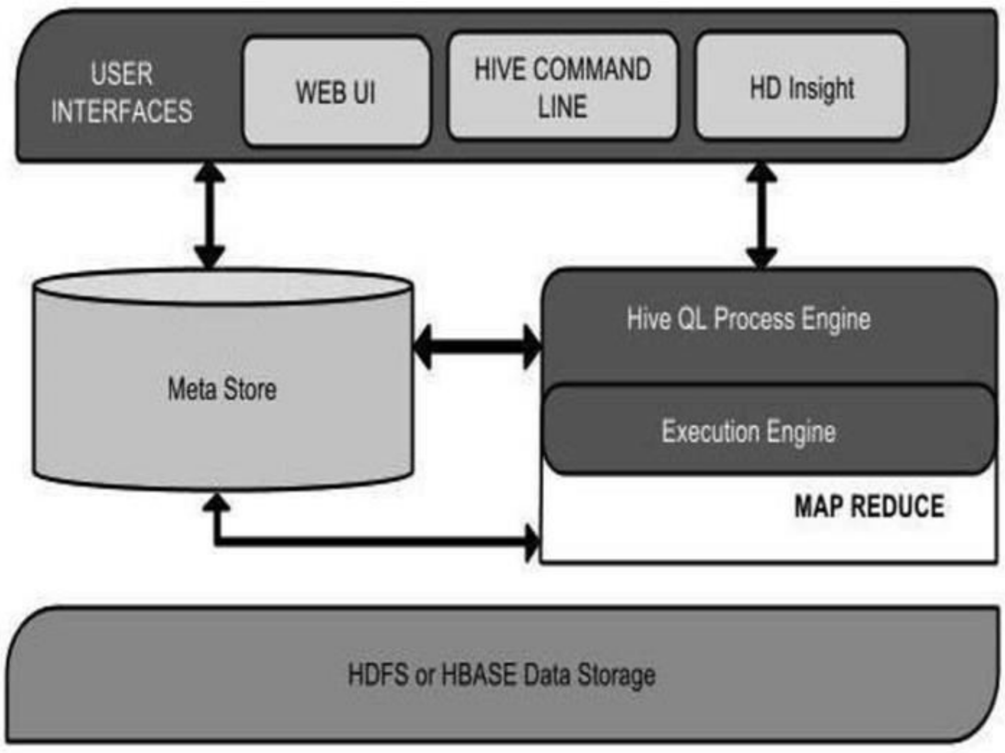
Comparison with traditional database

The storage and querying operations of Hive closely resemble those of traditional databases. While Hive is a SQL dialect, there are a lot of differences in structure and working of Hive in comparison to relational databases. The differences are mainly because Hive is built on top of the Hadoop ecosystem, and has to comply with the restrictions of Hadoop and MapReduce.

A schema is applied to a table in traditional databases. In such traditional databases, the table typically enforces the schema when the data is loaded into the table. This enables the database to make sure that the data entered follows the representation of the table as specified by the table definition. This design is called *schema on write*. In comparison, Hive does not verify the data against the table schema on write. Instead, it subsequently does run time checks when the data is read. This model is called *schema on read*. two approaches have their own advantages and drawbacks. Checking data against table schema during the load time adds extra overhead, which is why traditional databases take a longer time to load data. Quality checks are performed against the data at the load time to ensure that the data is not corrupt. Early detection of corrupt data ensures early exception handling. Since the tables are forced to match the schema after/during the data load, it has better query time performance. Hive, on the other hand, can load data dynamically without any schema check, ensuring a fas initial load, but with the drawback of comparatively slower performance at query time. Hive does have an advantage when the schema is not available at the load time, but is instead generated later dynamically.[22]

Transactions are key operations in traditional databases. As any typical RDBMS, Hive supports all four properties of transactions (ACID): Atomicity, Consistency, Isolation, and Durability. Transactions in Hive were introduced in Hive 0.13 but were only limited to the partition level.[26] Recent version of Hive 0.14 were these functions fully added to support complete ACID properties. Hive 0.14 and later provides different row level transactions such as *INSERT, DELETE and UPDATE*.[27] Enabling *INSERT, UPDATE, DELETE* transactions require setting appropriate values for configuration propertiessuch as hive.support.concurrency, hive.enforce.bucketing, and hive.exec.dynamic.partition.mode.[

Hive Architecture



Component diagram of hive architecture

User Interface :
Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).

Meta Store :

Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.

HiveQL Process Engine :

HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.

Execution Engine:

The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine.

Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.

HDFS or HBASE

Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

Metastore in hive:

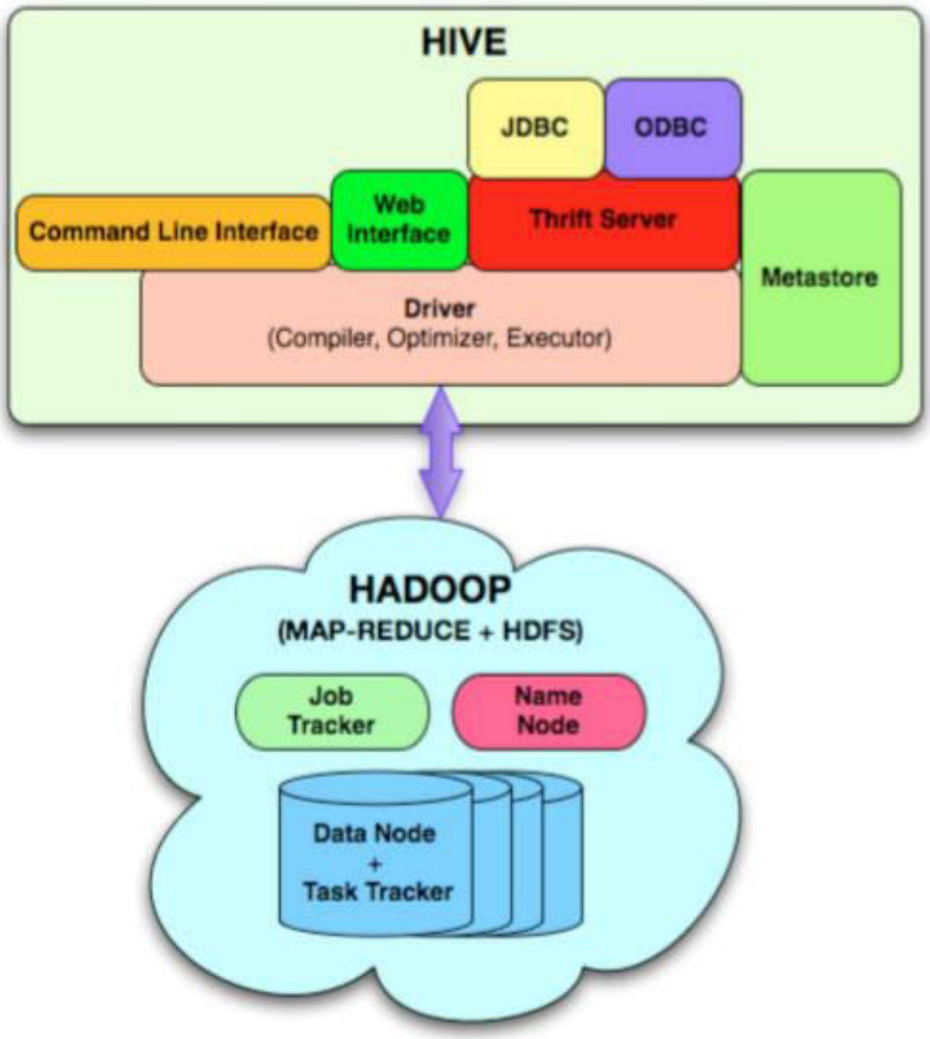
All the metadata for Hive tables and partitions are accessed through the Hive Meta store [4]. Derby is a default meta store in hive, but we replaced it by MYSQL. Ther are various advantage of MYSQL over Derby.

1. In MYSQL meta data is stored in database where as in Derby it is stored in file system.

2. In MYSQL, multipleuser can work in same metastore at a time, but it is not possible in case of derby.

3. In MYSQL, remote connection can be established, but in dery it is not possible.

In MYSQL, we can give permission to specific user to access, but in dery it is not possible



Hive system architecture and interaction with Hadoop

External Interfaces : Hive provides both user interfaces like command line (CLI) and web UI, and application programming interfaces (API) like JDBC and ODBC.

The Hive Thrift Server exposes a very simple client API to execute HiveQL statements. Thrift is a framework for cross-language services, where a server written in one language (like

Java) can also support clients in other languages. The Thrift Hive clients generated in different languages are used to build common drivers like JDBC (java), ODBC (C++), and scripting drivers written in php, perl, The Driver manages the life cycle of a HiveQL statement during compilation, optimization and execution. On receiving the HiveQL statement, from the thrift server or other interfaces, it creates a session handle which is later

used to keep track of statistics like execution time, number of output rows, etc.

- Execute Query : The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute.
- Get Plan : The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query.
- Get Metadata : The compiler sends metadata request to Metastore (any database).
- Send Metadata: Meta store sends metadata as a response to the compiler.
- Send Plan: The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.
- Execute Plan :The driver sends the execute plan to the execution engine.
- Execute Job: Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job.
- Metadata Ops: Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
- Fetch Result: The execution engine receives the results from Data nodes.
- Send Results :The execution engine sends those resultant values to the driver.
- Send Results: The driver sends the results to Hive Interfaces.

Hive Functionality

Hive Server is an API that enables the clients to process the queries on hive data warehouse and obtain the required results. Under hive server driver, compiler and execution engine communicate with each other and execute the query.

The client posts the query via a GUI. The driver accepts the queries in the first instance from GUI and it will define session handlers, which will fetch required APIs that is modelled with different interfaces like JDBC or ODBC. The compiler generates the plan for the job to be processed. Compiler in turn is in touch with matter and it obtains metadata from Meta Store.

Execution Engine is the vital component here to process a query by directly interacting with Job Tracker, Name Node and Data nodes. By running hive query at the backend, it will produce a series of Map Reduce Jobs. In this scenario, the execution engine acts like a bridge between hive and Hadoop to execute the query. For HDFS operations, Execution Engine communicates Name Node.

At the end, it is going to fetch required results from Data Nodes. It will be having duplex communication with Meta store.

Hive Data Modelling

Hive works with two types of table structures - internal and external, rely on the design of schema and how the data is getting placed into Hive

I. Internal Table:

- Create tables
- Load the data.

By dropping this type of table, both data and schema will be ousted. The stored location of this table will be at /user/hive/warehouse.

II. External Table:

Data will be obtainable in HDFS; the table is going to get created with HDFS data. By dropping the table, it removed only schema, data will be obtainable in HDFS as before. External tables give a choice to create multiple schemas for the data kept in HDFS instead of dropping the data every time whenever schema upgrades

The three main ***functions*** of Hive are:

- Data Summarization
- Querying
- Analysis

Some of the basic/main ***features*** of Hive are:

- It facilitates reading, writing and managing large datasets using SQL like language known as HiveQL.
- It eases the process of data summarization, querying and analysis.
- It provides indexes to accelerate queries.
- It is very scalable, extensible, familiar and fast.

Advantages

- It helps the programmers by reducing the need to write complex Map Reduce programs to process data using Hadoop.
- HiveQL is similar to SQL statements, thus easy to understand.
- It reuses the concept of relational database, such as, tables, rows, columns etc so that the learners can easily relate and understand.
- It is an efficient ETL tool.
- It supports all the client applications, written in Python, PHP, C++, Java or Ruby by exposing its thrift server.
- It performs analytics on large datasets and works well for complex queries.
- It supports Bitmap indexes
- It contains a range of user function APIs which can be used to build custom behaviour into the query engine.
- Higher level query language – Simplifies working with large amounts of data
- Lower learning curve than Pig or MapReduce – Less trial and error than Pig
- Hive is built on Hadoop, so supports and handles all the capablities of Hadoop provides like reliable, high available,node failure, commodatiy hardware
- Database developer need not to learn the java programming for writing map reduce programs for retrieving data from Hadoop system.
- Fits the lowslevel interface requirement of Hadoop perfectly.
- Supports external tables which make it possible to process data without actually storing in HDFS.
- It has a rule based optimizer for optimizing logical plans.
- Supports partitioning of data at the level of tables to improve performance.
- Metastore or Metadata store is a big plus in the architecture which makes the lookup easy.

Disadvantages

- It does not offer real-time queries.
- It does not offer row-level update
- Provides acceptable latency for interactive data browsing.
- Sub-queries are not supported in Hive
- Latency for Apache Hive queries is generally very high.
- Not designed for Online Transaction Processing
- Supports overwriting or apprehending data but not updates and deletes.
- It is not designed for Online transaction processing (OLTP), but is used for the Online Analytical Processing (OLAP).

- Hive supports overwriting or apprehending data, but does not support updates and deletes.

New Features in hive 2.0

- **Hbase to store Hive Metadata-** The current metastore implementation is slow when tables have thousands or more partitions. With Tez an Spark engines we are pushing Hive to a point where queries only take a few seconds to run. But planning the query can take as long as running it. Much of this time is spent in metadata operations [15].
- **Long-lived daemons for query fragment execution, I/O and caching** – LLAP is the new hybrid execution model that enables efficiencies across queries, such as caching of columnar data, JIT-friendly operator pipelines, and reduced overhead for multiple queries (including concurrent queries), as well as new performance features like asynchronous I/O, pre-fetching and multi-threaded processing
- **HPL/SQL – Implementing Procedural SQL in Hive** – In this new release we have PL/HQL tool (www.plhql.org) that implements procedural SQL for Hive (actually any SQL-on-Hadoop implementation and any JDBC source).
- **Hive on Spark Container** – When Hive job is launched by Oozie, a Hive session is created and job script is executed. Session is closed when Hive job is completed. Thus, Hive session is not shared among Hive jobs either in an Oozie workflow or across workflows.
- **Hive-on-Spark parallel ORDER BY** – This is the one of the greatest feature, as if we need to sort the records then we have to manually set / force the reducer count to 1 to have it in single file.
- **Dynamic Partition Pruning** – Tez implemented dynamic partition pruning and this is a nice optimization and we should implement the same in HOS.
- **Hive-on-Spark Self Union/Join** – A Hive query may try to scan the same table multi times, like self join, self-union, or even share the same subquery. As you may know that, Spark support cache RDD data, which mean Spark would put the calculated RDD data in memory and get the data from memory directly for next time, this avoid the calculation cost of this RDD at the cost of more memory usage.

Hive 2.1-25X faster

Interactive query with Hive LLAP. LLAP was introduced in Hive 2.0 and improved in Hive 2.1 to deliver 25x faster performance than Hive 1 (covered in detail below). Robust SQL ACID support with more than 60 stabilization fixes. 2x Faster ETL through a smarter CBO, faster type conversions and dynamic partitioning optimizations. Procedural SQL support, dramatically simplifying migration from EDW solutions. Vectorization support for text files, introducing an option for fast analytics without any ETL. A host of new diagnostics and monitoring tools including a new HiveServer2 UI, a new LLAP UI and an improved Tez UI.

Over all we conclude

Apache Hive is mainly used for data querying, analysis, and summarization. It helps improve developers' productivity which usually comes at the cost of increasing latency. Hive is a variant of SQL and a very good one indeed. It stands tall when compared to SQL systems implemented in databases. Hive has many user-defined functions that offer effective ways of solving problems. It is easily possible to connect Hive queries to various Hadoop packages like RHive, RHive, and even Apache Mahout. Also, it greatly helps the developer community work with complex analytical processing and challenging data formats.Hive allows users to simultaneously access data and, at the same time, increases the response time, i.e., the time a system or a functional unit takes to react to a given input. In fact, Hive typically has a much faster response time than most other types of queries. Hive is also highly flexible as more commodities can easily be added in response to adding more clusters of data without any drop in performance.

Conclusion

From this comprehensive literature review, we found that Hive is a higher level query language that simplifies working with large amounts of data. Also it has lower learning curve than Pig or MapReduce, the HiveQL is much closer to SQL than pig and Less trial and error than pig. Inspite of these advance features Hive has some setbacks too like u pdating data is complicated mainly because of using HDFS, it can add records and overwrite partitions too. Hive can access real time access to data and need to use other means like Hbase or Impala. Hive has high latency too.

References

1. Hive: A Literature Review
2. Apache Hive. Available at http://hive.apache.org.
3. Introduction and Performance: An Over view of Hive
4. Hive – A Petabyte Scale Data Warehouse Using
5. HIVE- Processing Structured Data in HADOOP
6. A Review on Hive and Pig
7. A Contemplating approach for Hive and Map reduce for efficient Big Data Implementation
8. Hive Performance Benchmark. Available at http://issues.apache.org/jira/browse/HIVE-396 .
9. Hive wiki at http://www.apache.org/hadoop/hive.