BigData Concepts

03 January 2024 09:51

Working with Hive QL

Lecture
- Datatypes,
- Operators and Functions,
- Hive Tables (Managed Tables and Extended Tables),
- Partitions and Buckets,
- Storage Formats,
- Importing data,
- Altering and Dropping Tables

Lab-Assignment:
- Creative a hive DB and table (internal and external)
- Load the data into hive table (using local inpath and HSFS inpath)

10. Querying with Hive QL

Lecture
- Querying Data-Sorting,
- Aggregating,
- Map Reduce Scripts,
- Joins and Sub queries,
- Views,
- Map and Reduce side joins to optimize query.

Lab-Assignment:
- Run all the types of joins in Hive
- Execute the data to be partitioned

More on Hive QL

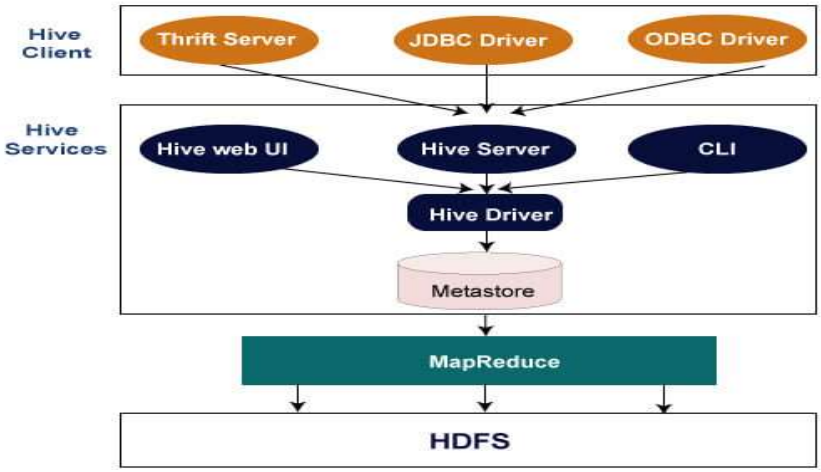
Lecture
- Data manipulation with Hive,
- UDFs,
- Appending data into existing Hive table,
- custom map/reduce in Hive
- Writing HQL scripts

Apache Hive

It is a data warehousing and SQL-like query language system built on top of Hadoop.
Provides a high-level interface for managing and querying large datasets - stored in a distributed storage system (HDFS)
Hive is a part of the Apache Hadoop project which is designed to enable data analysts and developers to easily analyze and process Structed and semi-structured data using SQL - like queries.

1. HiveQL (Hive Query language)
2. Hive Metastore
   - store metadata about hive tables, partitions and their associated schemas
   - It is separate from the query engine, allows multiple hive instances to share the same metadata
   - It contains information about the structure of the data, the location of the data files and other essentials metadata.
3. Hive Execution Engine
4. Hive UDF (User - Defined Functions)
   - Used to extend the functionality of hive for custom operations.
   - UDF are written in java or other supported languages
5. Hive Ser De (Serializer / Desterilized)
   - Define how data is serialized and deserialized when moving between hive and other external systems
   - It helps in interpreting the structure of data stored in various formats (eg:- JSON, CSV, Avro and many more)
6. HIVE CLI (command line interface)
7. HiveServer2
   - It allows external applications and tools to communicate with hive and submit queries
   - Provides thrift, JDBC & ODBC interface to Hive
8. Hive Web Interface
   - Web-based interface which allows interactions with hive services through graphical user interfaces
   - Default port is 10002

Hive Architecture



1. User Interface (CLI and Web Interface)
   - Hive CLI --> allows users to submit HiveQL queries, manage table and also perform Admin tasks
   - Web ---> Provides the facility for interaction with hive services through gui and used for visual management and querying
2. Drives
   - This receives the HiveQL queries submitted by the users through CLI or Web.
   - Responsible for processing and managing the execution of these queries
3. Compiler
   - The HiveQL queries are taken and compiles them into an execution plan.
   - Exection plan is a series of MapReduce jobs or tasks (other execution engine specific task) that will be used to process and analyze the data
4. MetaStore
   - Centralized repository - store metadata about hive tables, partitions and their associated schemas
   - It is separate from the query engine, allows multiple hive instances to share the same metadata
   - It contains information about the structure of the data, the location of the data files and other essentials metadata.
   - To allow multiple hive instances to share the same metadata the metastore is decoupled for the query engine
5. Execution Engine
   - Responsible for execution the complied execution plan generated by the compiler.
   - Hive uses MapReduce as one of the execution engine but we can use execution engines like Apache Spark, Apache Tez etc
6. Hive Server (HiveServer 2)
   - It allows external applications and tools to communicate with hive and submit queries
   - Provides thrift, JDBC & ODBC interface to Hive
   - Supports multi-client connectivity and hence used for remote connectivity and also provides concurrency
7. Hadoop Distributed File System (HDFS)
   - Hive stores data in HDFS, we can also use other distributed storage systems.
   - HDFS is the primary storage layer for large-scale data that hive processes and analyzed

The Hive architecture involves components for query submission, compilation, meta data storage, execution and interfaces for user interaction.

Comparing Hive database with Traditional Database

	Hive Database	**Traditional Database**
Query Language	• Hive QL – Hive Query Language – which is similar to SQL • The queries are translated into MapReduce jobs or other execution engines to process data stored in Hadoop	• SQL – Structured Query Language • Use for defining and manipulating the data and is a standard language used to interaction with any RDBMS .
Storage Format	• Hive Database (HDFS) • We can store in various formats (eg- text, JSON , Avro, etc..)	• Tables with fixed schema to store data. • Storage format depends on the database engine • We use the traditional file system provided by the OS
Schema	• Hive supports changes for the data schema over time without requiring modification of the existing data. Hence it supports schema evolution.	• Require careful management of schema changes and altering the schema – we require downtime and complex process involved and hence lot of procedures to be followed for migrating.
Data Model	• HiveQL follows a schema-on-read approach. (schema is applied when query the data rather than when it is ingested) • Supports semi-structured and structured data which is suitable for processing large scale , diverse data sets that are stored in HDFS	• This approach follows schema-on-write. (eg:- MsSQL, MSSQL, PostgreSQL, Oracle …) • Predefined schema is created before date is inserted into the database. • These databases are optima for structured data with a fixed schema.
Use Case	• Well suited for batch processing. ETL (Extract, transform, load) tasks, analytic queries on large volumes of data • Commonly used in big data processing workflows	• Well suited for transactional applications, real-time querying, where data structure is well-defined and data or the structure does not change frequently

NOTE: Installing Hive --> Refer to day 13 section pages for detailed procedure

$ schematool -dbType derby -initSchema

```
schematool -dbType derby -initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLo
ggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.35.jar!/or
g/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:        jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
Starting metastore schema initialization to 2.3.0
Initialization script hive-schema-2.3.0.derby.sql
Initialization script completed
schemaTool completed
hadoop@mainserver1:/usr/local/hive/conf$
```

root@mainserver1: /usr/local/hive/lib

```
hadoop@mainserver1:/usr/local/hive/conf$ vim hive-site.xml
hadoop@mainserver1:/usr/local/hive/conf$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLo
ggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.35.jar!/or
g/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in file:/usr/local/hive/conf/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a diff
erent execution engine (i.e. spark, tez) or using Hive 1.X releases.
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.hive.common.StringInternUtils (file:/usr/local/hi
ve/lib/hive-common-2.3.9.jar) to field java.net.URI.string
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.hive.common.StringInternU
tils
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
hive>
```

192.168.56.100 - PuTTY

```
hadoop@mainserver1:~$ jps
1175 Jps
hadoop@mainserver1:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [mainserver1]
2024-01-03 08:53:47,136 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platfor
m... using builtin-java classes where applicable
hadoop@mainserver1:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@mainserver1:~$
```

```
hadoop@mainserver1:/usr/local/hive/conf$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLo
ggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.35.jar!/or
g/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in file:/usr/local/hive/conf/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a diff
erent execution engine (i.e. spark, tez) or using Hive 1.X releases.
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.hive.common.StringInternUtils (file:/usr/local/hi
ve/lib/hive-common-2.3.9.jar) to field java.net.URI.string
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.hive.common.StringInternU
tils
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
hive> show databases;
OK
default
Time taken: 17.332 seconds, Fetched: 1 row(s)
hive>
```

Hive > create database cdacdb;
Hive> creaate databbase if not exists cdacdb;
Hive> show databases;
Hive> creaate databbase if not exists hivedb;
Hive> show databases;
Hive> use cdacdb;

```
hive> SET mapreduce.framework.name=local;
hive> use cdacdb;
OK
Time taken: 0.603 seconds
hive> insert into table dept values (2, 'shivaji' , 'maharashtra');
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider usi
ng a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hadoop_20240103112935_fa184a0c-4aa1-4d12-bcc4-f07ef0611e81
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2024-01-03 11:29:51,043 Stage-1 map = 100%,  reduce = 0%
Ended Job = job_local1573212769_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/cdacdb.db/dept/.hive-staging_hive_2024
-01-03_11-29-35_826_1029879613847936666-1/-ext-10000
Loading data to table cdacdb.dept
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 22 HDFS Write: 111 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 17.583 seconds
hive>
```