Using Hive for Data Warehousing: Introduction to Hive

1 Hi. Welcome to Accessing Hadoop Data Using Hive. This lesson will provide you with an Introduction to Hive.

2 After completing this lesson, you should be able to:
   Describe what Hive is, what it’s used for and how it compares to other similar technologies.
   Describe the Hive architecture.
   Describe the main components of Hive.
   And list interesting ways others are using Hive.

3 Hive was initially developed by Facebook in 2007 to help the company handle massive amounts of new data. At the time Hive was created, Facebook had a 15TB dataset they needed to work with. A few short years later, that data had grown to 700TB.
   Their RDBMS data warehouse was taking too long to process daily jobs so the company decided to move their data into the scalable open-source Hadoop.
   The company found that creating MapReduce programs was not easy and was time consuming for many users.
   When they created Hive, their vision was to bring familiar database concepts to Hadoop, making it easier for all users to work with.
   In 2008 Hive was open sourced. Facebook has since used Hive for reporting dashboards and ad-hoc analysis.

4 So what exactly is Hive? Hive is a data warehouse system built on top of Hadoop. Hive facilitates easy data summarization, ad-hoc queries, and the analysis of very large datasets that are stored in Hadoop.
   Hive provides a SQL interface, better known as HiveQL or HQL for short, which allows for easy querying of data in Hadoop. HQL has its own Data Definition and Data Manipulation languages which are very similar to the DML and DDL many of us already have experience with.
   In Hive, the HQL queries are implicitly translated into one or more MapReduce jobs, shielding the user from much more advanced and time consuming programming.
   Hive provides a mechanism to project structure (like tables and partitions) onto the data in Hadoop and uses a metastore to map file structure to tabular form.

5 Hive is not a full database. However Hive can fit right alongside your RDBMS. There are a variety of things that Hive lacks when compared to an RDBMS.
Hive is not a real-time processing system and is best suited for batch jobs and huge datasets. Think heavy analytics and large aggregations. Latencies are often much higher than in a traditional database system. Hive is schema on read which provides for fast loads and flexibility, at the sacrifice of query time. Hive lacks full SQL support and does not provide row level inserts, updates or deletes. Hive does not support transactions and has limited subquery support. Query optimization is still a work in progress too. If you are interested in a distributed and scalable data store that supports row-level updates, rapid queries and row-level transaction, then HBase is also worth investigating.

Let’s compare Hive to a couple of common alternatives. An example often used is that of the Word Count program. The Word Count program is meant to read in documents on Hadoop and return a listing of all the words read in along with the number of occurrences of those words. Writing a custom MapReduce program to do this takes 63 lines of code. Having Hive perform the same task only takes 7 easy lines of code! Another Hive alternative is Apache Pig. Pig is a high level programming language, best described as a “data flow language” and not a query language. Being a custom language means there is a higher learning curve for SQL programmers to become comfortable with the Pig language. Pig has powerful data transformation capabilities and is great for ETL. It is not so good for ad-hoc querying. Pig is a nice complement for Hive and the two are often used in tandem in a Hadoop environment.

Now let’s take a look at Hive’s architecture. There are a variety of different ways that you can interface with Hive.

You can use a web browser to access Hive via the Hive Web Interface.

You could also access Hive using an application over JDBC, ODBC, or the Thrift API, each made possible by Hive’s Thrift Server referred to as HiveServer. HiveServer2 was released in Hive 0.11 and serves as a replacement for HiveServer1, though you still have the choice of which HiveServer to run, or can even run them concurrently. HiveServer2 brings many enhancements including the ability to handle concurrent clients and more.

Hive also comes with some powerful Command Line interfaces (often referred to as the “CLI”). The introduction of HiveServer2 brings with it a new Hive CLI called Beeline, which can be run in embedded mode or thin client mode. In thin client mode, the Beeline CLI connects to Hive via JDBC and HiveServer2. The original CLI is also included with Hive and can run in embedded mode or as a client to the HiveServer1.

Hive comes with a catalog known as the Metastore. The Metastore stores the system catalog and metadata about tables, columns, partitions and so on. The metastore makes mapping file structure to a tabular form possible in Hive.
A newer component of Hive is called HCatalog. HCatalog is built on top of the Hive metastore and incorporates Hive's DDL. HCatalog provides read and write interfaces for Pig and MapReduce and uses Hive's command line interface for issuing data definition and metadata exploration commands. Essentially, HCatalog makes it easier for users of Pig, MapReduce, and Hive, to read and write data on the grid.

The Hive Driver, Compiler, Optimizer, and Executor work together to turn a query into a set of Hadoop jobs. The Driver piece manages the lifecycle of a HiveQL statement as it moves through Hive. It maintains a session handle and any session statistics. The Query Compiler compiles HiveQL queries into a DAG of MapReduce tasks. The Execution Engine executes the tasks produced by the compiler in proper dependency order. The Execution Engine interacts with the underlying Hadoop instance, working with the Name Node, Job Tracker and so on.

A typical Hive installation has the following directory structure. First there is a “lib” folder in the Hive installation. The lib folder contains a variety of JAR files. These JAR files contain the Java code that collectively make up the functionality of Hive. Then there is the “bin” directory. This is the location of a variety of Hive scripts that launch various Hive services. Finally there is the “conf” directory. This directory contains Hive’s configuration files.

Now let’s take a slightly deeper look at the Hive CLI. The CLI or “Command Line Interface” is the most common way to interact with the Hive system. From the CLI shell you can perform queries, DML, and DDL. You can view and manipulate table metadata, retrieve query explain plans, and more. Hive currently comes with two command line interfaces – the original CLI and the newer Beeline CLI. These two CLI’s are located in Hive’s bin directory. There are differences in the original CLI and Beeline architecture, however running commands within the two is a very similar procedure.

The metastore stores the Hive metadata. It consists of two pieces – the service and the datastore. There’s three configurations you can choose for your metastore. The first is embedded, which runs the metastore code in the same process with your Hive program and the database that backs the metastore is in the same process as well. The embedded metastore is likely to be used only in a test environment. The second configuration option is to run the metastore as local, which keeps the metastore code running in process, but moves the database into a separate process that the metastore code communicates with. The last option is to setup a remote metastore. This option moves the metastore itself out of the process as well. The remote metastore can be useful if you wish to share the metastore with other users. The remote metastore is the configuration you are most likely
to use in a production environment, as it provides some additional security benefits on top of what’s possible with a local metastore.
A minimum Hive configuration identifies where the metastore is located. If there are no configuration details provided by the user then an embedded Derby database is used. A Derby metastore only allows one user at a time, so it may be advantageous to setup Hive to use a more robust database option, such as DB2, MySQL or another JDBC-compliant database.

11
A good introduction to Hive has to include some cool real world examples of how companies are using Hive. Here’s a very small sampling of some Hive real world use cases.
As you can see, Hive is used for data mining, data analysis, analytics, customer facing business intelligence and a multitude of other uses.

12
You have now completed this topic. Thank you for watching.