Apache Hive

Apache Hive is a data warehouse software project built on top of Apache Hadoop for providing data query and analysis.[3] Hive gives an SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop. Traditional SQL queries must be implemented in the MapReduce Java API to execute SQL applications and queries over distributed data. Hive provides the necessary SQL abstraction to integrate SQL-like queries (HiveQL) into the underlying Java without the need to implement queries in the low-level Java API. Since most data warehousing applications work with SQL-based querying languages, Hive aids portability of SQL-based applications to Hadoop.[4] While initially developed by Facebook, Apache Hive is used and developed by other companies such as Netflix and the Financial Industry Regulatory Authority (FINRA).[5][6] Amazon maintains a software fork of Apache Hive included in Amazon Elastic MapReduce on Amazon Web Services.[7]

Apache Hive

Original author(s)	Facebook, Inc.
Developer(s)	Contributors (https://hive.apache.org/people.html)
Initial release	October 1, 2010[1]
Stable release	3.1.2 / August 26, 2019[2]
Repository	github.com/apache/hive (https://github.com/apache/hive)
Written in	Java
Operating system	Cross-platform
Available in	SQL
Type	Data warehouse
License	Apache License 2.0
Website	hive.apache.org (https://hive.apache.org)

Contents

Features

Apache Hive supports analysis of large datasets stored in Hadoop's HDFS and compatible file systems such as Amazon S3 filesystem and Alluxio. It provides a SQL-like query language called HiveQL[8] with schema on read and transparently converts queries to MapReduce, Apache Tez[9] and Spark jobs. All three execution engines can run in Hadoop's resource negotiator, YARN (Yet Another Resource Negotiator). To accelerate queries, it provided indexes, but this feature was removed in version 3.0 [10] Other features of Hive include:

- Different storage types such as plain text, RCFile, HBase, ORC, and others.

- Metadata storage in a relational database management system, significantly reducing the time to perform semantic checks during query execution.
- Operating on compressed data stored into the Hadoop ecosystem using algorithms including DEFLATE, BWT, snappy, etc.
- Built-in user-defined functions (UDFs) to manipulate dates, strings, and other data-mining tools. Hive supports extending the UDF set to handle use-cases not supported by built-in functions.
- SQL-like queries (HiveQL), which are implicitly converted into MapReduce or Tez, or Spark jobs.

By default, Hive stores metadata in an embedded Apache Derby database, and other client/server databases like MySQL can optionally be used.[11]

The first four file formats supported in Hive were plain text,[12] sequence file, optimized row columnar (ORC) format[13] and RCFile.[14] Apache Parquet can be read via plugin in versions later than 0.10 and natively starting at 0.13.[15][16] Additional Hive plugins support querying of the Bitcoin Blockchain.[17]

Architecture

Major components of the Hive architecture are:

- Metastore: Stores metadata for each of the tables such as their schema and location. It also includes the partition metadata which helps the driver to track the progress of various data sets distributed over the cluster.[18] The data is stored in a traditional RDBMS format. The metadata helps the driver to keep track of the data and it is crucial. Hence, a backup server regularly replicates the data which can be retrieved in case of data loss.
- Driver: Acts like a controller which receives the HiveQL statements. It starts the execution of the statement by creating sessions, and monitors the life cycle and progress of the execution. It stores the necessary metadata generated during the execution of a HiveQL statement. The driver also acts as a collection point of data or query results obtained after the Reduce operation.[14]
- Compiler: Performs compilation of the HiveQL query, which converts the query to an execution plan. This plan contains the tasks and steps needed to be performed by the Hadoop MapReduce to get the output as translated by the query. The compiler converts the query to an abstract syntax tree (AST). After checking for compatibility and compile time errors, it converts the AST to a directed acyclic graph (DAG).[19] The DAG divides operators to MapReduce stages and tasks based on the input query and data.[18]
- Optimizer: Performs various transformations on the execution plan to get an optimized DAG. Transformations can be aggregated together, such as converting a pipeline of joins to a single join, for better performance.[20] It can also split the tasks, such as applying a transformation on data before a reduce operation, to provide better performance and scalability. However, the logic of transformation used for optimization used can be modified or pipelined using another optimizer.[14]
- Executor: After compilation and optimization, the executor executes the tasks. It interacts with the job tracker of Hadoop to schedule tasks to be run. It takes care of pipelining the tasks by making sure that a task with dependency gets executed only if all other prerequisites are run.[20]
- CLI, UI, and Thrift Server: A command-line interface (CLI) provides a user interface for an external user to interact with Hive by submitting queries, instructions and monitoring the process status. Thrift server allows external clients to interact with Hive over a network, similar to the JDBC or ODBC protocols.[21]

HiveQL

While based on SQL, HiveQL does not strictly follow the full SQL-92 standard. HiveQL offers extensions not in SQL, including *multitable inserts* and *create table as select*. HiveQL lacked support for transactions and materialized views, and only limited subquery support.[22][23] Support for insert, update, and delete with full ACID functionality was made available with release 0.14.[24]

Internally, a compiler translates HiveQL statements into a directed acyclic graph of MapReduce, Tez, or Spark jobs, which are submitted to Hadoop for execution.[25]

Example

The word count program counts the number of times each word occurs in the input. The word count can be written in HiveQL as:[4]

```
DROP TABLE IF EXISTS docs;
CREATE TABLE docs (line STRING);
LOAD DATA INPATH 'input_file' OVERWRITE INTO TABLE docs;
CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
 (SELECT explode(split(line, '\s')) AS word FROM docs) temp
GROUP BY word
ORDER BY word;
```

A brief explanation of each of the statements is as follows:

```
DROP TABLE IF EXISTS docs;
CREATE TABLE docs (line STRING);
```

Checks if table `docs` exists and drops it if it does. Creates a new table called `docs` with a single column of type `STRING` called `line`.

```
LOAD DATA INPATH 'input_file' OVERWRITE INTO TABLE docs;
```

Loads the specified file or directory (In this case “input_file”) into the table. `OVERWRITE` specifies that the target table to which the data is being loaded into is to be re-written; Otherwise the data would be appended.

```
CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
(SELECT explode(split(line, '\s')) AS word FROM docs) temp
GROUP BY word
ORDER BY word;
```

The query `CREATE TABLE word_counts AS SELECT word, count(1) AS count` creates a table called `word_counts` with two columns: `word` and `count`. This query draws its input from the inner query `(SELECT explode(split(line, '\s')) AS word FROM docs) temp"`. This query serves to split the input words into different rows of a temporary table aliased as `temp`. The `GROUP BY WORD` groups the results based on their keys. This results in the `count` column holding the number of occurrences for each word of the `word` column. The `ORDER BY WORDS` sorts the words alphabetically.

Comparison with traditional databases

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*.[22] The 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 fast 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 had 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 properties such as `hive.support.concurrency`, `hive.enforce.bucketing`, and `hive.exec.dynamic.partition.mode`.[28]

Security

Hive v0.7.0 added integration with Hadoop security. Hadoop began using Kerberos authorization support to provide security. Kerberos allows for mutual authentication between client and server. In this system, the client's request for a ticket is passed along with the request. The previous versions of Hadoop had several issues such as users being able to spoof their username by setting the `hadoop.job.ugi` property and also MapReduce operations being run under the same user: hadoop or mapred. With Hive v0.7.0's integration with Hadoop security, these issues have largely been fixed. TaskTracker jobs are run by the user who launched it and the username can no longer be spoofed by setting the `hadoop.job.ugi` property. Permissions for newly created files in Hive are dictated by the HDFS. The Hadoop distributed file system authorization model uses three entities: user, group and others with three permissions: read, write and execute. The default permissions for newly created files can be set by changing the umask value for the Hive configuration variable `hive.files.umask.value`.[4]

See also

- Apache Pig
- Sqoop
- Apache Impala

- Apache Drill
- Apache Flume
- Apache HBase

References

1. "Release release-1.0.0 · apache/Hive" (https://github.com/apache/hive/releases/tag/release-1.0.0). *GitHub*.
2. "26 August 2019: release 3.1.2 available" (https://hive.apache.org/downloads.html#26-august-2019-release-312-available). Retrieved 28 August 2019.
3. Venner, Jason (2009). *Pro Hadoop* (https://archive.org/details/prohadoop0000venn). Apress. ISBN 978-1-4302-1942-2.
4. *Programming Hive [Book]* (https://www.safaribooksonline.com/library/view/programming-hive/9781449326944/).
5. Use Case Study of Hive/Hadoop (http://www.slideshare.net/evamtse/hive-user-group-presentation-from-netflix-3182010-3483386)
6. OSCON Data 2011, Adrian Cockcroft, "Data Flow at Netflix" (https://www.youtube.com/watch?v=Idu9OKnAOis) on YouTube
7. Amazon Elastic MapReduce Developer Guide (http://s3.amazonaws.com/awsdocs/ElasticMapReduce/latest/emr-dg.pdf)
8. HiveQL Language Manual (https://cwiki.apache.org/confluence/display/Hive/LanguageManual)
9. Apache Tez (http://tez.apache.org/)
10. Hive Language Manual (https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Indexing#LanguageManualIndexing-IndexingIsRemovedsince3.0)
11. Lam, Chuck (2010). *Hadoop in Action*. Manning Publications. ISBN 978-1-935182-19-1.
12. Optimising Hadoop and Big Data with Text and HiveOptimising Hadoop and Big Data with Text and Hive (http://www.semantikoz.com/blog/optimising-hadoop-big-data-text-hive/)
13. "ORC Language Manual" (https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC). *Hive project wiki*. Retrieved April 24, 2017.
14. "Facebook's Petabyte Scale Data Warehouse using Hive and Hadoop" (https://web.archive.org/web/20110728063630/http://www.sfbayacm.org/wp/wp-content/uploads/2010/01/sig_2010_v21.pdf) (PDF). Archived from the original (http://www.sfbayacm.org/wp/wp-content/uploads/2010/01/sig_2010_v21.pdf) (PDF) on 2011-07-28. Retrieved 2011-09-09.
15. "Parquet" (https://web.archive.org/web/20150202145641/https://cwiki.apache.org/confluence/display/Hive/Parquet). 18 Dec 2014. Archived from the original (https://cwiki.apache.org/confluence/display/Hive/Parquet) on 2 February 2015. Retrieved 2 February 2015.
16. Massie, Matt (21 August 2013). "A Powerful Big Data Trio: Spark, Parquet and Avro" (https://web.archive.org/web/20150202145026/http://zenfractal.com/2013/08/21/a-powerful-big-data-trio/). *zenfractal.com*. Archived from the original (http://zenfractal.com/2013/08/21/a-powerful-big-data-trio/) on 2 February 2015. Retrieved 2 February 2015.
17. Franke, Jörn (2016-04-28). "Hive & Bitcoin: Analytics on Blockchain data with SQL" (https://snippetessay.wordpress.com/2016/04/28/hive-bitcoin-analytics-on-blockchain-data-with-sql/).
18. "Design - Apache Hive - Apache Software Foundation" (https://cwiki.apache.org/confluence/display/Hive/Design). *cwiki.apache.org*. Retrieved 2016-09-12.
19. "Abstract Syntax Tree" (http://c2.com/cgi/wiki?AbstractSyntaxTree). *c2.com*. Retrieved 2016-09-12.

20. Dokeroglu, Tansel; Ozal, Serkan; Bayir, Murat Ali; Cinar, Muhammet Serkan; Cosar, Ahmet (2014-07-29). "Improving the performance of Hadoop Hive by sharing scan and computation tasks" (https://doi.org/10.1186%2Fs13677-014-0012-6). *Journal of Cloud Computing*. **3** (1): 1–11. doi:10.1186/s13677-014-0012-6 (https://doi.org/10.1186%2Fs13677-014-0012-6).
21. "HiveServer - Apache Hive - Apache Software Foundation" (https://cwiki.apache.org/confluence/display/Hive/HiveServer). *cwiki.apache.org*. Retrieved 2016-09-12.
22. White, Tom (2010). *Hadoop: The Definitive Guide* (https://archive.org/details/hadoopdefinitive0000whit). O'Reilly Media. ISBN 978-1-4493-8973-4.
23. Hive Language Manual (https://cwiki.apache.org/confluence/display/Hive/LanguageManual)
24. ACID and Transactions in Hive (https://cwiki.apache.org/confluence/display/Hive/Hive+Transactions)
25. "Hive A Warehousing Solution Over a MapReduce Framework" (https://web.archive.org/web/20131008161608/http://www.vldb.org/pvldb/2/vldb09-938.pdf) (PDF). Archived from the original (http://www.vldb.org/pvldb/2/vldb09-938.pdf) (PDF) on 2013-10-08. Retrieved 2011-09-03.
26. "Introduction to Hive transactions" (https://web.archive.org/web/20160903210039/http://datametica.com/introduction-to-hive-transactions). *datametica.com*. Archived from the original (http://datametica.com/introduction-to-hive-transactions/) on 2016-09-03. Retrieved 2016-09-12.
27. "Hive Transactions - Apache Hive - Apache Software Foundation" (https://cwiki.apache.org/confluence/display/Hive/Hive+Transactions#HiveTransactions-NewConfigurationParametersforTransactions). *cwiki.apache.org*. Retrieved 2016-09-12.
28. "Configuration Properties - Apache Hive - Apache Software Foundation" (https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties#ConfigurationProperties-hive.txn.manager). *cwiki.apache.org*. Retrieved 2016-09-12.

External links

- Official website (http://hive.apache.org)