HIVE-Processing Structured Data in Hadoop

Conference Paper · March 2017

CITATIONS
0

READS
1,457

1 author:

Anish Gupta
Amity University, Greater Noida Campus
15 PUBLICATIONS **19** CITATIONS

SEE PROFILE

HIVE- Processing Structured Data in HADOOP

[1]Anish Gupta, [2]Manish K. Gupta
Amity University, Gr. Noida
[1]agupta4@gn.amity.edu ,[2] mgupta@gn.amity.edu

***Abstract*—the size of data has been growing day by day in rapidly way. Using traditional approach, it make expensive to process large set of data. Hadoop is a popular framework written in java, being used by company like Yahoo, facebook, etc. to store and process large set of data on commodity hardware. To process structured types of data we used HIVE. Hive support a query processing like SQL called HiveQL. Facebook introduced HIVEQL. Hive also includes a system catalog - Metastore – that contains schemas and statistics, which are useful in data exploration, query optimization and query compilation [1]. Hive also includes a system catalog, Hive-Metastore, containing schemas and statistics, which is useful in data exploration and query optimization [2].**

Keywords: HIVEQL, Metastore, SQL

I. INTRODUCTION

The size of data sets being collected and analyzed in the industry for business intelligence is growing rapidly, making traditional warehousing solutions prohibitively expensive [2]. Hadoop is a popular open source frame work written in JAVA language to store and process huge set of that data. HIVE is a core component of HADOOP frame work. HIVE is case insensitive language. Hive is a data ware house and it store data in file format.

HADOOP has two components for storing the data and for processing the data.

1. HDFS (Hadoop Distributed File System)
2. MAPR (Map Reduce)

HDFS is used for storing the large set of data (Structure, semi structure and Unstructured).

MAPR is used for processing the large set of data.

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 [3].

The Compiler is invoked by the driver upon receiving a HiveQL statement. The compiler translates this statement into a plan which consists of a DAG of map reduce jobs.

The driver submits the individual map-reduce jobs from the DAG to the Execution Engine in a topological order. Hive currently uses Hadoop as its execution engine.

II. 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 [2].

- 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, ... >

III. HIVE ARCHITECTURE

Figure 1 shows the major components of Hive and its interactions with Hadoop. The main components of Hive are [2]:
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 di_erent 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.

Hive> load data localInPath '/home/hadoop/file.txt' into table students;

Select Command:
Hive> select *from students;

a. External Table:

CREATE EXTERNAL TABLE STUDENTS (roll number INT, name STRING, age INT, address STRING)
ROW FORMAT DELIMITED
FIELD TERMINATED BY ','
LOCATION ' '

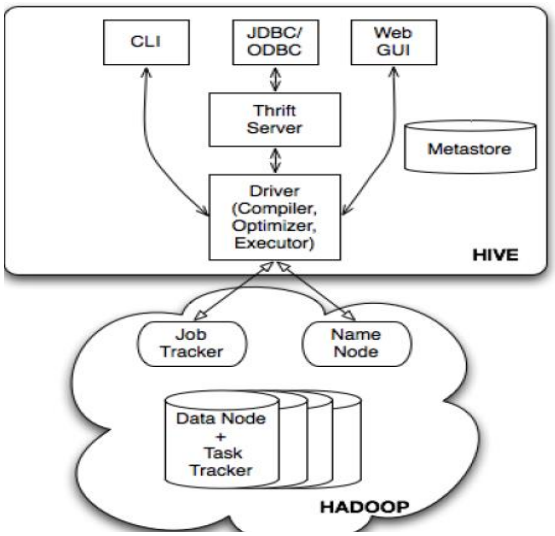

Fig 1.

IV. TABLE IN HIVE

There are 2 types of table in HIVE, Managed table and External table.

1.1 Managed Table: Managed table is like a normal database table where data can be stored and queried on. On dropping these table data also lost forever.

Creating an internal table:-

CREATE TABLE STUDENTS (roll_number INT, name STRING, age INT, address STRING)
ROW FORMAT DELIMITED
FIELD TERMINATED BY ',';

ROW FORMAT should have delimiters used to terminate the fields and lines like in the above example the fields are terminated with comma (",").

V. 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 . There 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, multiple user 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

VI. META STORE DEPLOYE MODEL

a. ***Embedded Model:***

Embedded mode is the default metastore deployment mode for CDH. In this mode, the metastore uses a Derby database, and both the database and the metastore service are embedded in the main HiveServer process. Both are started for you when you start the HiveServer process. This mode requires the least amount of effort to configure, but it can support only one active user at a time and is not certified for production use [5].

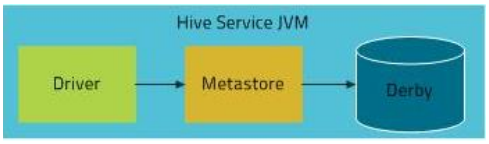

Fig. 2

b. Local Mode:

In Local mode, the Hive metastore service runs in the same process as the main HiveServer process, but the metastore database runs in a separate process, and can be on a separate host. The embedded metastore service communicates with the metastore database over

VII. 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 and

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 [15].
- **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) [15].
- **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 [15].
- **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 [15].
- **Dynamic Partition Pruning** – Tez implemented dynamic partition pruning and this is a nice optimization and we should implement the same in HOS [15].
- **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.

VIII. 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. [16]

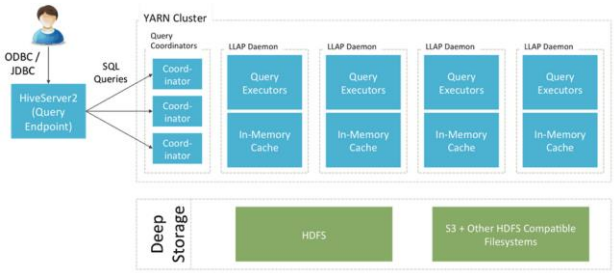

Fig. 3

REFERENCES

[1] Hive – A Petabyte Scale Data Warehouse Using
[2] Hadoop Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu and Raghotham Murthy.
[3] Hive A Warehousing Solution Over a MapReduce Framework Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff and Raghotham Murthy.
[4] https://www.tutorialspoint.com/hive/
[5] https://cwiki.apache.org/
[6] https://www.cloudera.com/
[7] *Venner, Jason (2009). Pro Hadoop. Apress. ISBN 978-1-4302-1942-2*
[8] *Lam, Chuck (2010). Hadoop in Action. Manning Publications. ISBN 1-935182-19-6*
[9] *Yongqiang He; Rubao Lee; Yin Huai; Zheng Shao; Namit Jain; Xiaodong Zhang; Zhiwei Xu. "RCFile: A Fast and Space-efficient Data Placement Structure in MapReduce-based Warehouse Systems"*
[10] *"Parquet". 18 Dec 2014. Archived from the original on 2 February 2015. Retrieved 2 February 2015*
[11] *Massie, Matt (21 August 2013). "A Powerful Big Data Trio: Spark, Parquet and Avro". zenfractal.com. Archived from the original on 2 February 2015. Retrieved 2 February 2015.*
[12] *Thusoo, Ashish; Sarma, Joydeep Sen; Jain, Namit; Shao, Zheng; Chakka, Prasad; Anthony, Suresh; Liu, Hao; Wyckoff, Pete; Murthy, Raghotham (2009-08-01).*

"Hive: A Warehousing Solution over a Map-reduce Framework". *Proc. VLDB Endow.* **2**

[13] "Abstract Syntax Tree". *c2.com*. Retrieved 2016-09-12.

[14] *Dokeroglu, Tansel; Ozal, Serkan; Bayir, MuratAli; Cinar, MuhammetSerkan; Cosar, Ahmet (2014-07-29).* "Improving the performance of Hadoop Hive by sharing scan and computation tasks". *Journal of Cloud Computing.* **3** *(1): 1–11.* doi:10.1186/s13677-014-0012-6

[15] "Hive Transactions - Apache Hive - Apache Software Foundation". *cwiki.apache.org*. Retrieved 2016-09-12.

[16] http://dataottam.com/2016/03/06/10-new-exciting-features-in-apache-hive-2-0-0/

[17] https://hortonworks.com/blog/announcing-apache-hive-2-1-25x-faster-queries-much/