Big Data Analytics Tools: A Review

V. P. Mahatme
KITS, Ramtek Dt. Nagpur, India
e-mail: mahatme.vilas@gmail.com

Yogeshwary Sarode
KITS, Ramtek Dt. Nagpur, India
e-mail: sarode.yogi@gmail.com

Shital Radke
KITS, Ramtek Dt. Nagpur, India
e-mail: shtlradke@gmail.com

***Abstract*—** The way of dealing with the data is changing rapidly. Making traditional warehousing solutions prohibitively expensive. In big data analytics, heterogeneity, scale, timeliness and day to day changing nature of data are the main concerns. There are various open-source implementations which are being used for shifting their analytical databases from high-end proprietary machines to cheaper, lower-end, commodity hardware. In this paper, we mainly focus on different tools like Hive, Hadoop, Cheetah which support the concept of big data analytics. It is also focused with the needs, architecture and applications big data analytics. Hadoop is an open source version of MapReduce framework. It is basically used to connect multiple databases. Hive is built on top of Hadoop. Hive supports queries expressed in a SQL-like declarative language called HiveQL, which are compiled into map reduce jobs that are executed using Hadoop. Cheetah is designed specifically for online advertising application to allow various simplifications and custom optimizations.

Keywords*-big data, mapreduce,hadoop,hive,cheetah*

*****

I. INTRODUCTION

The amount of data in world has been blowing up. Companies capture trillions of bytes of information about their customers, suppliers and operations. Millions of networked sensors are being embedded in the devices such as mobile phones and automobiles which are sensing, creating, and communicating data. Individuals with smart phones on social network sites are also contributing to this exponential growth. Big data is large pools of data that can be captured, communicated, aggregated, stored, and analyzed. Most research into big data has been focused on the question of its volume [1]. The Big Data phenomenon is related to the open source software revolution and deals with Hadoop, Hive, cheetah and other related software. These tools have been developed to aggregate, manipulate, analyze, and visualize big data. These tools support to several technique and technologies including statistics, computer science, applied mathematics, and economics. These tools used as an alternative to store and process extremely large data sets on commodity hardware. We present different tools of big data analytics. Hadoop is an open source software framework for processing huge datasets on certain kinds of problems on a distributed system. Hive, an open-source data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. It Uses MapReduce for query execution. Cheetah is another data warehouse tool, built on top of MapReduce. Cheetah is designed specifically for online advertising applications. It allows various simplifications and custom optimizations.

II. BACKGROUND AND OVERVIEW OF TOOLS

In this section, overview of the Hadoop, Hive and Cheetah tools of a big data analytics is mentioned.

A. Hadoop

The Hadoop framework is designed to provide a reliable, shared storage and analysis infrastructure to the user community [5]. The storage portion of the Hadoop framework is provided by a distributed file system solution as Hadoop Distributed File System (HDFS). Analytical functionality is presented by MapReduce. Several other components are part of the overall Hadoop solution suite. The MapReduce functionality is designed as a tool for deep data analysis and the transformation of very large data sets. Hadoop enables the users to explore and analyze complex data sets by utilizing customized analysis scripts and commands. In other words, via the customized MapReduce routines, unstructured data sets can be distributed, analyzed, and explored across thousands of shared processing systems called nodes. Hadoop's HDFS replicates the data onto multiple nodes to safeguard the environment from any potential data-loss.

B. Hadoop Subprojects

Hadoop is in general best known for the MapReduce and the HDFS components, there are several other subprojects as a unit. Hadoop is basically designed to efficiently process very large data volumes by linking many commodity systems together to work as a parallel entity.

Hadoop Common: The common utilities that support the other Hadoop modules.

HDFS: A distributed file system that provides high-throughput access to application data.

Hadoop YARN: A framework for job scheduling and cluster resource management.

Hadoop MapReduce: A YARN-based system for parallel processing of large data sets.

Avro: A data serialization system.

Cassandra: A scalable multi-master database with no single points of failure.

Chukwa: A data collection system for managing large distributed systems.

HBase: A scalable, distributed database that supports structured data storage for large tables.
Hive: A data warehouse infrastructure that provides data summarization and ad hoc querying.
Mahout: A Scalable machine learning and data mining library.
Pig: A high-level data-flow language and execution framework for parallel computation.
ZooKeeper: A high-performance coordination service for distributed applications.

C. Hadoop Database

In this section, hadoop database system and its components are described. Main goal of HadoopDB is to accommodate very large amounts of data and to provide high-throughput access to the data sets. Based on the HDFS design, the files are redundantly stored across multiple nodes to ensure high-availability for the parallel applications.

D. Hadoop Data Distribution

In a Hadoop cluster environment, the data is distributed among all the nodes during the data load phase. The HDFS splits large data files into chunks that are managed by different nodes in the cluster. Each chunk is replicated across several nodes to address single node outage or fencing scenarios. An active monitoring system re-replicates the data during node failure events. Despite the fact that the file chunks are replicated and distributed across several nodes, Hadoop operates in a single namespace and hence, the cluster content is collectively accessible. In the Hadoop programming framework, data is conceptually record-oriented. The individual input files are carved up into lines or other specific formats. Hence, each thread executing on a cluster node processes a subset of the records. The Hadoop framework schedules these threads in proximity to the location of the data/records by utilizing knowledge obtained from the distributed file system. Which data chunk is operated on by a node is chosen based on the data chunks locality to a node. The design goal is most data is read from a local disk straight into the CPU subsystem, to economize on the number of network transfers necessary to complete the processing cycle. The design strategy of moving the actual computation to the data allows Hadoop to achieve high data locality reference values that result in increased performance scenarios. HDFS loads two types of data Grep data and User Visit. Grep data uses unique key to identify different data. User Visit data includes repartitioning of data which contain URL and visiting date during analyzing the data. Initially each node contains scattred User Visit data, then HDFS partition data into blocks and stores it on individual node. The maximum node load time is same as the entire load time for both Grep and User Visits. Hadoop and hadoopDB works in combination and supportive to each other. hadoopDB mainly support parallel DBMS and Hadoop used for analyzing quires.

E. Hadoop Architecture

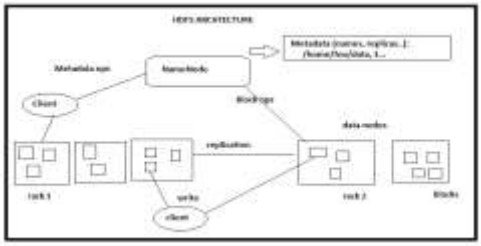

Figure1: HDFS Architecture

An HDFS cluster encompasses two types of nodes Name and DataNodes that operate in a master slave relationship. The NameNode reflects the master, while the DataNodes represent the slaves. The NameNode manages the file system namespace, maintains the file system tree, as well as the metadata for all the files and directories in the tree. All this information is persistently stored on a local disk via two files that are labeled the namespace image and the edit log, respectively. The NameNode keeps track of all the DataNodes where the blocks for a given file are located. That information is dynamic and not persistently stored, as it is reconstructed every time the system starts up. Any client can access the file system on behalf of a user task by communicating with the NameNode and the DataNodes respectively. The clients basically represent a POSIX like file system interface, so that the user code's functionalities do not require any actual knowledge about the Hadoop Name and DataNodes. The DataNodes store and retrieve blocks based on requests made by the by clients or the NameNode, and they do periodically update the NameNode with lists of the actual blocks that they are responsible for. Without an active NameNode, the file system is considered non-functional. Hence, it is paramount to safeguard the NameNode by insuring that the node is resilient to any potential failure scenarios.

III. HIVE

Hive is a data warehousing solution which is built over Hadoop. It is powered by HiveQL which is a declarative SQL language compiled directly into Map Reduce jobs which are executed over the underlying Hadoop architecture. Hive also allows the users to customize the HiveQL language and allows them to write queries which have custom Map Reduce code [7]. In this paper, the structure of Hive database, architecture of Hive and its flow along with the components which make up hive are also covered.

A. Hive Database

In Hive data is organized into Tables, Partitions, Buckets.

- Tables: In Hive, the basic structure in the database is a table, analogous to the tables in relational database. All the tables in hive database have an associated HDFS directory. Tables which are external to Hive and lie in a different file system like NFS or local directories are also supported.
- Partitions: Each table can have one or more partitions in Hive. The data is distributed within the Hadoop File System under sub directories.
- Buckets: The data in partitions is further distributed as buckets. The division in buckets is based on the hash of column in a table. Each bucket is stored as a file in the partition sub-directories.

B. HiveQL

Hive QL is a SQL like language for handling data warehousing using Hive over Hadoop. Queries like union, aggregation and join, select queries, and sub queries are supported.

HiveQL allows creation of new tables in accordance with the partitions and buckets and also allows insertion of data in single or multiple tables but does not allow deletion or updating of data. Another feature unique to HiveQL is multi-table insert. In this construct, users can perform multiple queries on the same input data using a single HiveQL query. Hive optimizes these queries to share the scan of the input data, thus increasing the throughput of these queries several orders of magnitude.

C. Hive Architecture

Figure shows the major components of Hive and its interactions with Hadoop. The main components of Hive are:

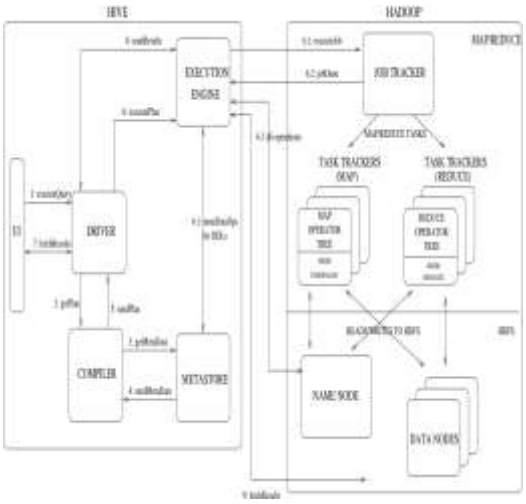

Figure2: HIVE Architecture

- UI – The user interface for users to submit queries and other operations to the system.
- Driver – The component which receives the queries. This component implements the notion of session handles and provides execute and fetch APIs modeled on JDBC/ODBC interfaces.
- Compiler – It parses the query does semantic analysis on the different query blocks and query expressions and eventually generates an execution plan with the help of the table and partition metadata looked up from the Metastore.
- Metastore – It stores all the structure information of the various tables and partitions in the warehouse including column and column type information. The serializers and deserializers are necessary to read and write data and maintain corresponding HDFS files where the data is stored.
- Execution Engine – It executes the execution plan created by the compiler. The plan is a DAG of stages. The execution engine manages the dependencies between these different stages of the plan and executes these stages on the appropriate system components.

Figure2 shows how a typical query flows through the system. The UI calls the execute interface to the Driver. The Driver creates a session handle for the query and sends the query to the compiler to generate an execution plan. The compiler gets the necessary metadata from the Metastore. This metadata is used to typecheck the expressions in the query tree as well as to prune partitions based on query predicates. The plan generated by the compiler is a DAG of stages with each stage being either a map/reduce job, a metadata operation or an operation on HDFS. For map/reduce stages, the plan contains map operator trees and a reduce operator tree. The execution engine submits these stages to appropriate components. In each task (mapper/reducer) the deserializer associated with the table or intermediate outputs is used to read the rows from HDFS files and these are passed through the associated operator tree. Once the output is generated, it is written to a temporary HDFS file though the serializer (this happens in the mapper in case the operation does not need a reducer). The temporary files are used to provide data to subsequent map/reduce stages of the plan. For DML operations the final temporary file is moved to the table's location. This scheme is used to ensure that dirty data is not read (file rename being an atomic operation in HDFS). For queries, the contents of the temporary file are read by the execution engine directly from HDFS as part of the fetch call from the Driver.

IV. CHEETAH

Cheetah is known as data warehouse system which is developed on the top on the MapReduce. It is mainly designed for online application handling. It helps to design complex query languages [3]. Cheetah tool uses map reduce up to its full extent. It is based on relation database and data warehouse concepts. It is developed in cluster format. Cheetah provides powerful but simple query language which can be efficiently used on client side. It provides effective data compression technique based on Hadoop.

A. Cheetah Architecture

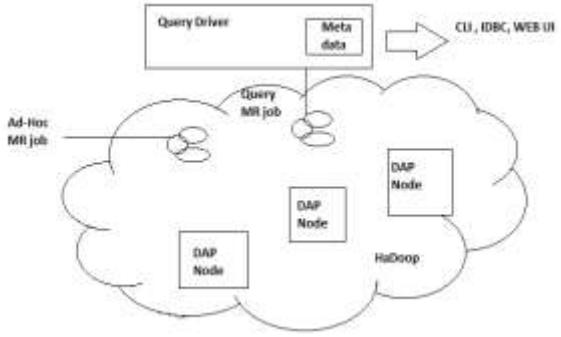

Figure3: Cheetah Architecture

It provides simple interface to implement MapReduce. User can interact with the system by using interface such as web user interface or JDBC or ODBC.

Main task of Query Driver is to submit query to the data node on which query driver is running. It translate query to map reduce job which consist of map function and reduce function.

DAP node consist of data access primitives which act as interface between nodes and users. It acts as a scanner for virtual view.

- Query Execution

  Cheetah works on MapReduce and Hadoop. All the input files are stored on hadoop file system. Input file is taken from HDFS in the form of fact tables and submitted to MapReduce. Then MapReduce automatically consider map function and reduce function. Map function and reduce function will execute the dimensions. If output of any queries is big in the terms of tables then MapReduce function is used to reduce it. Speed of execution of queries is 10 times faster than other tools.

- Integration

  Cheetah provides interactive interfaces such as JDBC and web user interfaces. It uses low CPU overhead. It provides good response to the multi queries. It provides structured query languages interface to MapReduce for handling complex queries.

V. CONCLUSION

All the big data analytical tools are open source. Hadoop provides parallel query executing system. It can handle fault tolerance efficiently. Its attractive feature is capacity of handling heterogeneous data and dealing large amount of data for query optimization. Hive system fits the low level interface requirement of Hadoop perfectly. It 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. Hive supports partitioning of data at the level of tables to improve performance. In Hive, Metastore or Metadata store is a big plus in the architecture which makes the lookup easy. There are certain disadvantages of hive such as No support for update and delete. It does not support for singleton inserts. Data is required to be loaded from a file using LOAD command. No access control implementation. In hive Correlated sub queries are not supported. Cheetah provides abstract view which is conceded as central point in designing of this tool. Multi queries can be efficiently executed by using map reduce function. Cheetah can be developed as cluster of data. This tool uses MapReduce in combination with Hadoop for better query optimization.

REFERENCES

[1] Wei Fan & Albert Bifet, "Mining Big Data: Current Status, and Forecast to the Future", Vol. 14, Issue 2, pp.1-5, Dec.2012.

[2] "Challenges and Opportunities with Big Data", A community white paper developed by leading researchers across the United States accessed on http://www.cra.org/ccc/files/docs/init/bigdatawhitepaper.pdf

[3] Songting Chen., "Cheetah: A High Performance, Custom Data Warehouse on Top of MapReduce", Vol.3, Issue1-2, pp.1459-1468 Sept.2010.

[4] Hadoop HDFS User Guide at http://hadoop.apache.org accessed on Sept.2014.

[5] Hadoop Map-ReduceTutorial at http:// hadoop.apache.org accessed on Sept.2014.

[6] HadoopDBProject at http://db.cs.yale.edu/hadoopdb/hadoopdb.html

[7] Hive Language Manual at http://wiki.apche.org/hadoop/Hive