www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 11 Nov. 2016, Page No. 18790-18792

Scalability Study of Hadoop MapReduce and Hive in Big Data Analytics

Khadija Jabeen[1], Dr TSS Balaji[2]
[1]B.tech, Dept. of Computer Science and Engineering, Brindavan Institute of Technology and Science, Affiliated to JNTUA, Peddatekur, Kurnool-518218, India
khadijajabeen68@gmail.com
[2]Principal, Brindavan Institute of Technology and Science, Affiliated to JNTUA, Peddatekur, Kurnool-518218, India
tssbalaji@gmail.com

Abstract: ***Hadoop is a data management solution for the analysis of Big Data. In Hadoop, Hive is used to store the metadata. This study compares the scalability of Hadoop MapReduce and Hive for small and medium datasets besides showing how the metadata can be created, loaded, accessed and stored using Hive – a data warehousing solution built on top of Hadoop. To make the comparison of scalabilities Hadoop MapReduce and Hive, a word count program was investigated using two data management solutions- Hadoop MapReduce and Hive. This comparison demonstrates that the Hadoop MapReduce programming model is very low level and it will make the developers write custom programs which are hard to maintain and reuse, where as Hive uses an SQL-like query language called HiveQL to store large amounts of data consuming less time and also plugs in the Map Reduce scripts into queries.***

Keywords: Hadoop, Hadoop Map Reduce, Hive, metadata, HiveQL, HDFS.

1. Introduction

The major search engines and ecommerce companies started wrestling with ever-growing quantities of data from the early days of the Internet's mainstream breakout. In the recent past, social networking sites experienced the same problem. Today, many organizations realize that the data they gather is a valuable resource for understanding their customers, the performance of their business in the marketplace, and the effectiveness of their infrastructure. [1]

The Hadoop ecosystem emerged as a cost-effective way of working with such large datasets. It imposes a particular programming model, called MapReduce, for breaking up computation tasks into units that can be distributed around a cluster of commodity, server class hardware, thereby providing cost-effective, horizontal scalability [3]. Underneath this computation model is a distributed file system called the Hadoop Distributed File System (HDFS). Although the file system is "pluggable," there are now several commercial and open source alternatives.

However, a challenge remains; how do you move an existing data infrastructure to Hadoop, when that infrastructure is based on traditional relational databases and the Structured Query Language (SQL)? This is where Hive comes in. Hive provides an SQL dialect, called Hive Query Language (abbreviated HiveQL or just HQL) for querying data stored in a Hadoop cluster.

Hive is best suited for data warehouse applications, where a large data set is maintained and mined for insights, reports, etc. Because most data warehouse applications are implemented using SQL-based relational databases, Hive lowers the barrier for moving these applications to Hadoop. Hive makes it easier for developers to port SQL-based applications to Hadoop, compared with other Hadoop languages and tools. [1]

2. Hive

While the MapReduce framework provides scalability and low-level flexibility to run complex jobs on large data sets, it may take several hours or even days to implement a single MapReduce job. Recognizing this, Facebook developed Hive based on familiar concepts of tables, columns and partitions, providing a high-level query tool for accessing data from their existing Hadoop warehouses [4]. The result is a data warehouse layer built on top of Hadoop that allows for querying and managing structured data using a built on top of Hadoop that allows for querying and managing structured data using a familiar SQL-like query language, HiveQL, and optional custom MapReduce scripts that may be plugged into queries. Hive converts HiveQL transformations to a series of MapReduce jobs and HDFS operations.

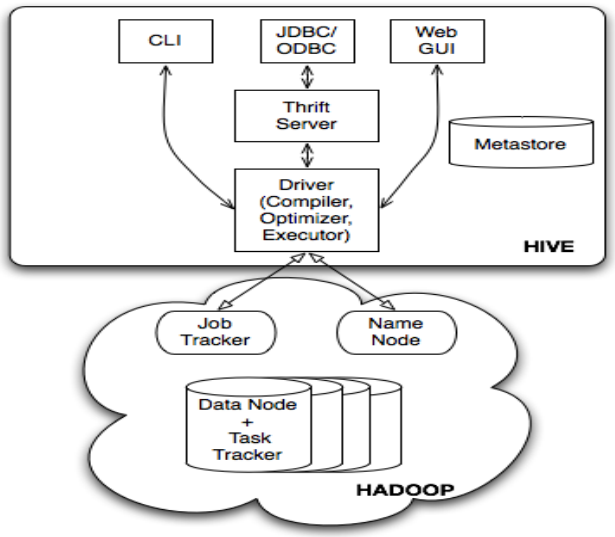

Figure 1: Hive Architecture

The Hive data model is organized into tables, partitions and buckets. The tables are similar to RDBMS tables and each corresponds to an HDFS directory. Each table can be divided into partitions that correspond to sub-directories within an HDFS table directory and each partition can be further divided into buckets which are stored as files within the HDFS directories .It is important to note that Hive was designed for

scalability, extensibility, and batch job handling, not for low latency performance or real-time queries. Hive query response times for even the smallest jobs can be of the order of several minutes and for larger jobs.

Because Hadoop HDFS has its own way of storing records that is in flat files in key value pair, there had to be an interface above it which would allow users to query Hadoop by using a language similar to SQL, this job is done by the interface HIVE. "Hive is a data warehouse system for Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in Hadoop compatible file systems. [2]

3. MapReduce

MapReduce is a programming model on top of HDFS for processing and generating large data sets which was developed as an abstraction of the map and reduces primitives present in many functional languages. The abstraction of parallelization, fault tolerance, data distribution and load balancing allows users to parallelize large computations easily. The map and reduce model works well for Big Data analysis because it is inherently parallel and can easily handle data sets spanning across multiple machines. Each MapReduce program runs in two main phases: the map phase followed by the reduce phase.

***Map Phase*.** The input to the map phase is the raw data. A map function should prepare the data for input to the reducer by mapping the key to the value for each "line" of input. The key-value pairs output by the map function are sorted and grouped by key before being sent to the reduce phase.

***Reduce Phase*.** The input to the reduce phase is the output from the map phase, where the value is an iterable list of the values with matching keys. The reduce function should iterate through the list and perform some operation on the data before outputting the final result. [1,2]

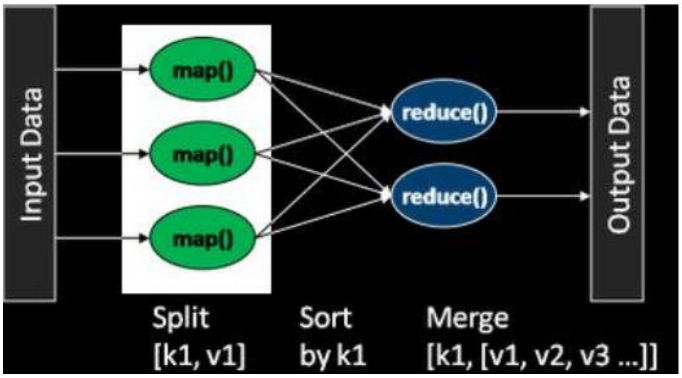

Figure 2: Representation of Map and Reduce functions

Map Function

```
map(input_record){
...
emit(k1,v1)
...
emit(k2,v2)
...
}  (1)
```

Reduce Function

```
reduce(key,values){
while(values.has_next){
aggregate=merge(values.next)
}
collect(key,aggregate)
}  (1)
```

4. Hive vs. Regular MapReduce

The size of data sets being collected and analyzed in the industry for business intelligence is growing rapidly, making traditional warehousing solutions prohibitively expensive. Hadoop is a popular open-source map-reduce implementation which is being used as an alternative to store and process extremely large data sets on commodity hardware. However, the map-reduce programming model is very low level and requires developers to write custom programs which are hard to maintain and reuse. Whereas Hive is an open-source data warehousing solution built on top of Hadoop. Hive supports queries expressed in a SQL-like declarative language - HiveQL, which are compiled into map-reduce jobs executed on Hadoop. In addition, HiveQL supports custom map-reduce scripts to be plugged into queries.

Hive-Metastore, contains schemas and statistics, which are useful in data exploration and query optimization. In Facebook, the Hive warehouse contains several thousand tables with over 700 terabytes of data and is being used extensively for both reporting and ad-hoc analyses by more than 100 users.

5. Comparison of Hive and MapReduce

Hadoop MapReduce is a framework for processing large data sets in parallel across a Hadoop cluster. Data analysis uses a two-step map and reduce process. The top level unit of work in MapReduce is a job. A job usually has a map and a reduce phase.

The **Hive** data model is organized into tables, partitions and buckets. The tables are similar to RDBMS tables and each corresponds to an HDFS directory. Each table can be divided into partitions that correspond to sub-directories within an HDFS table directory and each partition can be further divided into buckets which are stored as files within the HDFS directories. It is important to note that Hive was designed for scalability, extensibility, and batch job handling, not for low latency performance or real-time queries.

5.1 Wordcount using Hive

In hive first we need to create a table for that and load data in the form of text file into that table. Hive will perform MapReduce job internally and it will display the data as follows:

hadoop 2
world 2

Hive performs the MapReduce job internally and it has taken "35.456 seconds" of time to give the above output. The output is the count of words in the given input text file.

5.2 Wordcount using MapReduce

Consider **wordcount** as an example that will compare the performance issues of both mapreduce and hive jobs. The word count operation takes place in two stages a mapper phase and a reducer phase. In mapper phase first the test is tokenized into words then we form a key value pair with these words where the key being the word itself and value '1'. For example consider the sentence**:**

"**hadoop world"**>file1
"**hadoop world"**>file2

In map phase the sentence would be split as words and form the initial key value pair as

<hadoop,1>
<world,1>
<hadoop,1>
<world,1>

In the reduce phase the keys are grouped together and the values for similar keys are added. So here there are two pair of similar keys 'hadoop' and 'world' the values for these keys would be added so the output key value pairs would be

<hadoop,2>
<world,2>

MapReduce takes 1minute and 10.26 seconds to give the output as count of words that are given in the input.

Finally, MapReduce has taken 1 min and 10 seconds to give the output which is more than the time taken by hive i.e; "35.456" seconds. This shows that Hive has surpassed the MapReduce performance.

6. Conclusion

It can be observed from the word-count program performance comparison between Hive and MapReduce that, Hive performance remained constant and better for all sizes of data, matching and surpassing MapReduce performance specially, in case of larger data sets and can be concluded that the scalability of Hive is very high.

References

[1] Edward Capriolo, Dean Wampler, and Jason Ruthergl, "Programming Hive", Tata Mc GrawHill, 1992.

[2] Tom White, "Hadoop the Definitive Guide", 3rd Edition, O'Reilly Media, 2012.

[3] Apache Software Foundation, "Hadoop Releases," apache.org, Dec. 10, 2011. [Online]. Available: http://en.wikipedia.org/wiki/Apache_Hadoop. [Accessed: Dec. 06, 2014].

[4] Apache Software Foundation, "Apache Hive", Hadoop Releases, apache.org, Dec. 10, 2011. [Online]. Available: http://en.wikipedia.org/wiki/Apache_Hive.[Accessed: Mar. 09, 2016].

[5] Dean, Jeffrey, Ghemawat, Sanjay. "MapReduce: Simplified Data Processing on Large Clusters", OSDI, 2004.