International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Index Copernicus Value (2016): 79.57 | Impact Factor (2017): 7.296

BigData: A Case Study of Spark Mllib and Hive

Shubhajoy Das[1]

Mtech[IT],MBA, Agartala, Ramnagar-7, Pin:799002, India
shubhajoy.das@gmail.com

Abstract: ***The extent to which data is generated has shown a tremendous increase in the past decade because of social networks, sensor networks, geographic information systems, Financial Institutions, Supply chains. The storage capacities of computers have increased to stay competitive, but a big problem is that the access speeds of the disk has not improved to that extent to be at par with disk space improvement. Big Data comes to the rescue with a framework to analyse massive amounts of data in a distributed environment which is both horizontally and vertically scalable. Data sets with trillions of rows can be analysed very fast to provide valuable insights from data. Cloud service providers such as Amazon, Alibaba Cloud have made available robust infrastructure for Big Data. We study Apache Hive, Spark Mllib in profiling a Social Network Dataset and Collaborative Filtering algorithm in Spark Mllib for movie recommendations.***

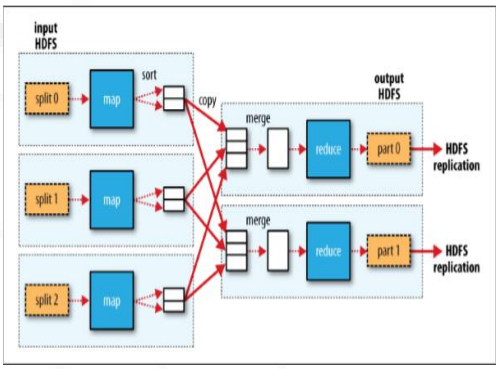
Keywords: BigData, SparkMllib, Collaborative Filtering, Hadoop, Spark, Apache, Hive, Amazon aws, HDFS

1. Introduction to Hadoop and its Components

Hadoop is based on a paper on Google big query. Hadoop works upon the concept of mapper and reducer and works on structured, semistructured and unstructured data. We create a large number of mapper and reducer classes. Hive is a subset of Structured Query Language which can be used to process trillions of rows and provide valuable insights about data. Big Data is characterized by 3 Vs of big data they are volume, velocity and veracity. Hadoop runs on java in two modes they are pseudo distributed mode, distributed mode. Hadoop has a name node, data node and secondary name node. Name node contains the metadata of tables and secondary name node is the fail over node.

Spark is a framework which runs on top of Hadoop and is 10 times faster than Hadoop. It has a Machine Learning module which supports large number of algorithms which can work on Big Data. It can persist data in resilient distributed data sets and data frames and perform computations on them. Spark can use Spark streaming for streaming in Amazon Aws and Microsoft Azure provides us with cloud infrastructure where we can launch a large number of nodes and perform our computations. Spark has become the industry standard for Big Data processing and has significant performance gains over its competitors such as Apache mahout. Amazon emr can be used to launch Hadoop clusters and run Spark programs. Amazon EC2 medium cluster with a single node has been configured which has 16gb of Ram and spark and hive components have been installed. Hadoop and its components require a huge amount of heap space so ec2 small instances run into jvm out of memory error. Amazon provides us a service Amazon EMR where we can launch a multi node Spark Cluster and run Spark jobs.

2. Map / Reduce



Reference: Hadoop the definite guide, Tom White

Hadoop deals with unstructured and semistructured data which are not defined in a schema so <Key, Value> pairs are used and a programming paradigm known as map/reduce is used. The input data is first split into Mappers and they are shuffled, sorted, merged and combined.

a. Apache hive

Apache hive is a data warehouse software product built on top of Apache Hadoop for providing data query and analysis. It allows users to ingest and analyse large amounts of data in a distributed environment using SQL. Apache hive uses java based derby database by default. Hive uses MAP/REDUCE for execution, HDFS for storage. Hive is used extensively by **Facebook** tasks such a spam detection, Data Mining, Summarization, Ad analysis. Hive supports avro and parquet format for serializing data when inserting in hive. Hive query language does not support the full SQL-92 specification. When you import data into hive from an external source there are two ways hive can store the data, It can store it in a managed table in hive warehouse directory or use an external table to specify which directory data will be stored. Hive database works on top of the MAP/REDUCE Framework. Hive has options to partition tables based on a specific column which improves the query processing time. It allows a dozen built in functions. Hive also supports bucketing a table. Bucketing is a phenomenon where a single table is divided into a large number of buckets which allows querying on a

subset of the table and also aids in sampling of data. Hive works on a JDBC type 4 driver.

Apache avro is a data serialization framework which stores data in json and in a compact binary representation. Apache parquet format is format which stores data in a column oriented way so that they are all adjacent. When querys are executed I/O is optimized.

Querys run on social network dataset in Apache Hive

An Amazon ec2 instance with Hadoop was configured which had 16gm of RAM and is t2.medium instance. The following querys were executed to get valuable insights from a social network dataset. The query execution performance can be optimized if we use a multi node Hadoop cluster. We use Hadoop in a pseudo distributed mode.

Create table for data ingestion

Create external table social (question_id string, user_id int, question_score int, question_time string, tags array<string>, question_views string, answer_count int, answer_id int, answer_user_id int, answer_score int, answer_time string, p string) row format delimited fields terminated by '_' collection items terminated by ', ' stored as textfile location '/user/shubhajoydas_rocketmail/shubhajoy';

The top 10 most used tags

select d.f, d.g from (select count(c.c1) as f, c.c1 as g from social as s LATERAL VIEW EXPLODE(s.tags) c as c1 group by c.c1) as d order by d.f desc limit 10

Questions answered within the past 1 hour

```
Selectv.d, v.q
from(selecthour(from_unixtime(cast(substr(s.answer_time, 0, length(s.answer_time)-
1)asbigint)))hour(from_unixtime(cast(s.question_time as bigint)))as w, day(from_unixtime(cast(substr(s.answer_time, 0, length(s.answer_time)-1)asbigint)))-
day(from_unixtime(cast(s.question_time as bigint)))
as w1, c.c1 as d, s.question_id as q
from social as s LATERAL VIEW EXPLODE(s.tags) c as c1) as v where v.w=0 and v.w1=0 limit 30
```

3. Collaborative Filtering

Collaborative Filtering is a technique where a user is provided with recommendations based on his preferences. There are various techniques in collaborative filtering such as cosine similarity, jaccard distance.

a. Cosine similarity

$Cos(\Theta)=A.B/||A||.||B||=\sum Ai.Bi/\sqrt{\sum Ai^2}.\sum Bi^2$
Cosine similarity uses the vector space model in which some attributes are chosen and similarity is quantified in the range 0 to 1.

b. Jaccard distance

$JD=|A\cap B|/|A\cup B|$
Matrix factorization is a technique where the coocurrence matrix contains <userid, movieid, rating> data and it is divided into two matrices which may contain unfilled values and the Alternating Least Squares Algorithm is used to fill the incomplete data. Spark-Mllib has a robust implementation for Alternating Least Squares algorithm which is scalable.

c. SPARK

Spark is a framework for big data processing having support for Machine Learning, SQl, Big Data streaming. It can run in the cloud, standalone or in a local cluster and has support for multiple operating systems .It is 100 times faster than hadoop. It can ingest data from a large number of formats. The working unit for Spark are resilient distributed Datasets and Data Frames. Spark can run on java, python, scala. Spark Runs jobs in stages and an RDD lineage is followed that an application particular task occurs only before a specific operation.

We can persist an RDD when a number of Mapper tasks are being run. DataFrames have column names and it is easier to use dataFrames because of the naming convention.

Collaborative Filtering pyspark implementation

```
import time
from pyspark.mllib.recommendation import ALS, MatrixFactorizationModel, Rating
from pyspark.sql import SQLContext, Row
tags=sc.textFile("ml-20m/tags.csv")
genomescores=sc.textFile("ml-20m/genome-scores.csv")
genometags=sc.textFile("ml-20m/genome-tags.csv")
links=sc.textFile("ml-20m/links.csv")
movies=sc.textFile("ml-20m/movies.csv")
ratings=sc.textFile("ml-20m/ratings.csv")

def cleanse(tags):
 tags=tags.split(', ')
 return tags
tag1=ratings.first()
ratings=ratings.filter(lambda tags:tags!=tag1)
ratings=ratings.map(cleanse)
tags1=tags.first()
tags12=tags.filter(lambda tags:tags!=tags1)
tags12=tags12.map(cleanse)
tags13 = tags12.map(lambda p: Row(userId=int(p[0]), movieId=int(p[1]), tag=int(p[2]), timestamp=p[3]))
tags12.take(5)
def typecast(tags):
 return Rating(tags[0], tags[1], tags[2])
tc1=ratings.map(typecast)
ty=tc1.take(10000)
ty=sc.parallelize(ty)
rank = 5
numIterations = 10
ty.cache()
traindata, testdata_1=ty.randomSplit([0.80, 0.20], 0)
model = ALS.train(traindata, rank, numIterations, 0.01, 1)
```

```
testdata = testdata_1.map(lambda p: (p[0], p[1]))
testdata.cache()
movies=model.recommendProducts(int(testdata.first()[0]), 5)
for x in range(0, 5):
 print(movies[x])
predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
ratesAndPreds = testdata_1.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
predictions.map(lambda r: ((r[0][0], r[1]))).take(50)
MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
print("Mean Squared Error = " + str(MSE))
tr=ratesAndPreds.join(tags)
```

We used the mean square error metric

MeanSquaredError= 0.7415299864531355

$MSE=\sum(x_{exp-Xobs)2/n}$

4. Spark Mllib Algorithms

Spark is a very robust framework and there are a huge number of parameters to tune which increases with the number of clusters and algorithms used. The Spark Mllib has algorithms for clustering, classification, regression and for creating feature vectors from text. Spark deep learning module is also present which uses large layers of neural networks to solve problems.

a>Logistic Regression

$\log(p(x)/1-p(x))=f(x)=w_{0+}w_1x+w_2x+w_3x$The equation above specifies for a particular data item, described by feature vector x,
The log-odds of the class equal to our linear function $P_+(x)=1/(1+e^{-x})$

Spark has a Logistic Regression Algorithm which can be used to classify multiclass data.

b>Support Vector Machines

Support Vector Machines is a classification technique which uses kernels to find a seperating hyperplane. The hyperplane can be found out by using various search techniques such as gradient descent.SVM has been extensively used in Face Recognition. Spark has an implementation of SVM.

b>Naïve Bayes Classifier

Bayes Classifier is a multiclass classification technique which uses Bayes theorem for classifying the data. In Spark Mllib naïve Bayes can be used through the mllib.classification. NaiveBayes class. Tuning a Spark cluster and parameters for performance.

When we are running operations on RDDS which join two or more RDDS there is no specific logic how Sparks core places data in different mappers or how they are combined. If we use a<Key, Value> pair programming model its very important on how the data is partitioned in different mappers.

There are different types of partitioning available in Spark

- Hash partitioning: It uses the Object.hashCode() method of java to partition the data into buckets.
- Range partitioning: The data in a particular range are placed in buckets

When we launch a spark job we can specify the number of threads to use. Apache Spark allows us to serialize data in a number of formats including json, textfile.

Spark streaming is a very robust way of spark to stream data and it is fault tolerant. It can run for 24 hours

Persisting Spark Data:

Level	Space Used	CPU time	In Memory	On Disk
Memory_only	High	Low	Y	N
Memory_only_ser	Low	High	Y	N
Memory_and_disk	High	Medium	Some	Some
Memory_and_disk_ser	Low	High	Some	Some
DISK_ONLY	Low	High	N	Y

Memory_And_Disk level splits to disk if there is too much data to fit in memory

Spark Mllib has a pipelining api similar to scikit-learn pipelining which helps us in selecting the best parameters for our machine learning algorithms.

5. Running External Scripts from Spark

When we write a script in R programming language and would like to take the advantage of the distributed capabilities of Spark we can load a R script in Spark and run our machine learning algorithms.

The output from our R script is passed as a pipeline to our Pyspark program.

6. Conclusion

Hadoop and Map/Reduce has provided us a new paradigm of computation which can be utilised for making algorithms faster. How the data in a Hadoop cluster is split can be manipulated to design better algorithms. Cloudera has sorted a large amount of numbers using Hadoop map reduce.

Hive and Hbase have become the industry standard for big data processing applications. Hadoop has support for Nosql databases such as Cassandra. Cloudera provides a distribution where we can configure a large number of nodes on Amazon Web Services.

References

[1]. Holden Karau, Andy Konwinski, Patrick Wendell, Matei Zaharia “Learning Spark Lightning Fast Data Analysis. Oreilly, pp. 213–239
[2]. Hadoop The definitive guide, 3rd ed.. Oreilly Yahoo Press: Tom White, pp.198-207.

[3]. Sandy Ryza, Uri Laserson, Sean Owen, Josh Wills "Advanced Analytics with Spark, " Orielly.
[4]. http://spark.apache.org/docs/latest/ml-guide.html.
[5]. https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/SingleCluster.html.
[6]. https://www.ibm.com/developerworks/library/ba-sentiment-analysis-big-data/index.html.
[7]. Collaborative Filtering Recommender Systems By Michael D. Ekstrand, John T. Riedl and Joseph A. Konstan, Foundation and trends in Human Computer Interaction.