Advances in Computer Science and Information Technology (ACSIT)
Print ISSN: 2393-9907; Online ISSN: 2393-9915; Volume 2, Number 4; April-June, 2015 pp. 321-325

http://www.krishisanskriti.org/acsit.html

Query Optimization in Hive for Large Datasets

Barkha Jain[1] and Manish Km. Kakhani[2]

[1]M-Tech student, Mody University, Lakshmangarh, Rajasthan
[2]Faculty of Engineering and Technology, Mody University, Lakshmangarh, Rajasthan
E-mail: [1]barkha.rawanka@gmail.com, [2]manishkakhani@gmail.com

Abstract—*Query optimization is growing research area in data analysis research. Database Management System strives to process the query in most efficient way to produce the best answer. Modern database technology uses Hive for analysis of large dataset. Hive is a framework for querying unstructured data as if it were structured by using Map-Reduce for processing and HDFS for storage. Most of the existing query optimization in Hive is related to shuffling, submitting and scheduling of jobs. We aimed to explore a suitable way of calculating large dataset on various parameters that is easily scalable both in response time and cost.*
In this paper, we develop an approach for optimizing HiveQL queries executed on the Hive framework. For this purpose, we implemented several operations such as Indexing, Normal view, Lateral view, an extension to Hive MetaStore on column statistics and Join ordering algorithms which are adapted to the specific needs of the Hadoop mapreduce environment to improve the overall performance of HiveQL query execution on MovieLen dataset collected from GroupLen website.

Keywords: *Big Data, Query Optimization, Hive, Indexing, Column Statistics, Map Join*

1. INTRODUCTION

Big Data is the term used for the collection of dataset so large and complex that it becomes difficult to process using traditional database management or processing tools. Gartner defines Big Data in terms of 3 V's that is Volume, Velocity and Variety, that are the assets require for processing data [1]. Additionally, there are more V's in the literature such as Validity, Veracity, Visibility, Value added by some researchers to explain Big Data more clearly [2]. There are varieties of applications and tools developed by various organizations to process and analyze Big Data. Hadoop is an open source Map- reduce project funded by Yahoo, emerged in year 2006 is used to process Exabyte or Zettabyte of data on cluster of commodity hardware connected by ethernet cables [3][4]. Hive is a data warehousing tool of Apache Foundation built on top of Hadoop distributed framework used to process and query data which is stored in HDFS in a similar manner as of traditional database management system (RDBMS). Hive initially developed by Facebook, is now used and developed by other companies such as Netflix [5]. As Hadoop distributed file solution has been the best solution for parallel, batch processing and aggregating flat files, Hive also uses HDFS for storage and offers convenient data retrieval to users as if they were using traditional database engine [6]. Hive uses Derby Language (No-RDBMS schema) to process unstructured data as if it were structured. Hive process data and stores in forms of tables and partitions, which can be accessed using a Hive specific query language called HiveQL similar to SQL which is easily handled by the people familiar with traditional database management system.

Since Hive is relatively young project, query optimization is a topic that comes into focus because Hive is still not in a stable state. On the Apache website [7], all recent releases are available which can be downloaded from mirrors, not necessarily in a stable state. As a result developers are enhancing the performance of Hive at this product development phase. Performance of any database engine can be measured by its response time and amount of work done by it. Hive version 0.13.1 is highly scalable and reliable as it uses basis Map-Reduce for processing. Many developers uses Hive because of its high response time. At the same time, this language also allows programmers who are familiar with the MapReduce framework to plug their own mappers and reducers programs to perform more sophisticated analysis that may not be supported by the built-in capabilities of the language.

In this paper, we are working on anonymized MovieLen dataset which was collected by the GroupLen Research Project at the University of Minnesota from the website [8]. This dataset consists of: 100,000 ratings (1-5) from 943 users on 1682 movies. Each user has rated at least 20 movies. The data was collected through the MovieLen website (movielens.umn.edu) during the seven-month period from September 19th, 1997 through April 22nd, 1998. The datasets consists information (age, gender, occupation, zip) of the users. Neither the University of Minnesota nor any of the researchers involved can guarantee the correctness of the data, its validity, and suitability; it completely depends upon its use. Using this datasets, we will describe implementation of various operations such as Indexing, Lateral view, Column Statistics, Map-Join, Drop and Truncate table; we will afterwards briefly explain how Column Statistics and Map-Join help in query optimization. At last we will conclude our

findings by suggesting the limitations and future scope in this area.

2. RELATED WORK

Due to tremendous growth of unstructured data generated from sensors, social media posts, purchase transactions records, cellphone conversations etc,there is a need of modern database system like Hadoop and Hive arises to process and analyze some data. Several research papers had been published in the area of query optimization in Hive. Anja Gruenheid et al. explained the concept of Query Optimization using Column Statistics in Hive. This paper implemented column level metadata that has been extracted according to the user and discuss various issues of cost. These column level statistics are then used with Join ordering Algorithm which further improves the overall performance of HiveQL query execution. With Hive it is possible to store the data in tables or partition which afterwards can be retrieved by Hive specific query language called HiveQL. The current version of Hive support Create and Drop Table on top of Hadoop distributed file system whereas Hive does not support Update and Delete Functionality up till date. Performance of Hive can be measured in terms of response time and amount of work it can do, no cross-product can be done in Hive which is commonly supported in Database Engine like Oracle or MySql. Column Level Statistics provides a query optimizer which provides selectivity of Join query which is further used in estimating actual table size. Even Rule Based optimization component can also be implemented in current version of Hive [6]. Apichon Witayangkurn et al. proposed an idea to implement spatial data on Hive over Hadoop [9].In this paper Spatial data processing is done on dataset collected from about 1.5 million users in Japan over 1 year period. Total no. of records is 9.2 billion and about 600GB in size and data files are kept in CSV format separated one files per day. Various papers proposed the idea of spatial dataset [10] [11].Yin hue et al. proposed significant improvements on storage efficiency and query execution plan [12]. It aims on maximizing effective storage capacity by providing access to data warehouse for various file formats as RCFile, ORCFile, CSVFile and effective cluster resource utilization and runtime performance of HIVE .It identified several shortcomings like in providing faster response time to the user coming from outside and low latency response in Hive. First shortcoming is Storage Efficiency of Hive because of One-row-at –a-time serialization prevents data values being efficiently compressed. Second shortcoming is data reading efficiency which is limited by the lack of indexes and non-decomposed columns with lack of indexes. Third shortcoming is query translation approach which ignores translation between data operations. Fourth shortcoming is that the runtime execution efficiency is limited by the one row at a time execution model. Storage efficiency is determined in Hive by SerDe which serializes every row into plain text or binary file sequence when writing data to HDFS and also deserializes these two formats back into rows with their original schema .Avrilia Floratou et al. compared the NOSQL system with the SQL system on the basis of performance, scalability aspects and workloads of the two systems i.e decision support analysis and interactive data serving [13]. Previously RDBMS is the only data driven application but now a day's newer NOSQL Database System such as MongoDB, CouchDB are popular alternatives to traditional RDBMS and can be used to process large dataset of Exabyte or Zettabyte of size. These NOSQL model is specially designed to work on key-value pair and especially on Cluster environment.Ashish Thusoo et al. reviewed a paper [14] showing larger datasets was also very useful in analyzing many Facebook site features. These features range from simple posting a post or comment to friend recommendations; it also says that Scribe, Hadoop and Hive together form the cornerstone of Facebook Infrastructure. The Data Flow Architecture of Facebook shows that how data is collected from various sources, different applications and different machines collectively stored in Scribe-Hadoop Clusters which is further extended in Production of Hive-Hadoop Cluster, Hive Replication is done in Adhoc Hive-Hadoop Cluster .There are two sources of Data –the federated MYSQL tier contains all the Facebook type related data and the web tier that generates all the log data. The federated MYSQL tier corresponds to dimension data and the data coming from the web servers corresponds to fact data. Many different components join together to give comprehensive platform to the user using Facebook. Future requirement in this infrastructure is to meet with the tremendous amount of data growth.

3. EXPERIMENTS

In this section we first presented the experimental setup. Second, we have shown the various implementations on Hive.

3.1. Experimental Setup

We desired to evaluate three approaches: HDFS, Map-Reduce Platform, and Hive Framework to run HiveQL query. For our experiment, we have used 6 systems of min 512MB RAM, storage capacity as 10GB as per our requirement, 8-Cores Xeon processors with 20M Cache, 2.90 GHz speed, 8.00 GT/s QPI speed, 2 QPI links and Red Hat Linux 6.0 64-bit operating system.

For Hadoop we use 6 systems to make a cluster with same specification as above consisting of 1 Client who assigns a job, 1 Namenode, 1 Jobtracker and remaining 3 are the Datanodes. LAN connected systems providing static IP is needed to form a cluster and Gigabit switch was used between cluster nodes. In total, there are 24 processing cores, however we set number of concurrent task to 7 tasks for each node because 1 core was reserved for other process therefore only (3*7 cores) 21 concurrent tasks can be done.

We use Hive Version 0.13.1 on the Hadoop version 1.2.1; we used Java Version 1.5.0 and JDK version 1.7 in our metrics.

We used default 64 MB HDFS block size and default replication factor was set to 3. The maximum Hadoop heap size used is 500 MB in the task.

We used MovieLen dataset consisting of 100000 ratings from 943 users on 1682 movies. Each user has rated at least 20 movies. The dataset used was anonymized in order to maintain the privacy of the users.

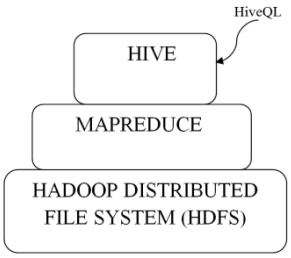

Fig. 1: Fig. of a Metrics

3.2. Query Implementation on Hive

In our research work several evaluation approaches were implemented on Hive. They are-

3.2.1. Indexing: Indexing improve the query lookups speed on certain columns of the table [15]. Suppose we want to find the movie whose category is comedy, for this query will predict as "where movie_category = comedy". This will process entire table or partitions or process all the rows. But if index exists for the category, then only that partition or row of the table will be processed which contains category as comedy.Hive indexing was added in version 0.7.0 and bitmap indexing was added in version 0.8.0.

Here pageads is the database name and genres, movies, item are the name of tables and genre_index, movies_index and item_index are name of their respective indexes.

```
hive>  create index genre_index on table genre (categories_id) as
       'BITMAP' WITH DEFERRED REBUILD;
       OK
       Time taken: 0.429 seconds

hive>  show tables;
       OK
       Pageads
       Genre
       pageads_genre_genre_index
       Time taken:0.03 seconds, Fetched : 3 row(s)

hive>  desc pageads__genre_genre_index__
       categories_id int
       _bucketname string
       _offset   int
       bitmaps  array<bigint>
       Time taken: 0.157 seconds, Fetched : 4 row(s)

hive>  create index movies_index on table movies (rating) as
       'COMPACT' WITH DEFERRED REBUILD;
       OK
       Time taken:0.736 seconds

hive>  show index on movies
       movies_index        movies   rating
       _movies_movies_index__                compact
       Time taken: 0.235 seconds, Fetched : 1 row(s)

hive>  desc pageads__movies_movies_index__;
       OK
       rating  int
       _bucketname string
       _offsets  array<bigint>
       Time taken: 1.305 seconds, Fetched : 3 row(s)

hive>  create index item_index on table item (movie_title) as
       'COMPACT' STORED AS RCFILE;
       OK
       Time Taken: 0.213 seconds

hive>  desc pageads_item_item_index__;
       OK
       movie_title  string
       _bucketname string
       _offsets  array<bigint>
```

3.2.2. Truncate and Drop table: In Hive 0.7.0 or later, DROP returns an error if the table doesn't exist unless IF EXIST is specified [16].

```
hive>  truncate table movies;
       OK
       Time taken: 1.613 seconds

hive>  Drop table movies;
       OK
       Time taken: 8.069 seconds
```

3.2.3. Column Statistics: Column Statistics help the Hive server query optimizer to determine the most efficient query plans. Column Statistics helps the query optimizer to choose a better plan in specific cases like involving multiple tables or the tables which already have Indexes [17].

```
hive>  select count (userid) as dist_ count, max(rating) as max,
       min (rating) as min, sum(rating) as sum_value from
       movies1;
       Total Map Reduce CPU Time Spent: 18 seconds 170
       msec
       OK

       100000  5       1       352986
       Time taken: 164.584 seconds, Fetched: 1 row(s)
```

3.2.4. Map Join: Hive does not support non equal join conditions as it is very difficult to express such conditions as a map/reduce job. Also, more than two tables can be joined in Hive [18].

```
hive>  select movies.itemid, movies.rating, users.age,
       users.occupation
       From movies
       INNER JOIN users ON movies.userid=users.userid;
```

Itemid	rating	Age	occupation
120	1	23	Librarian
762	2	28	Programmer
874	3	24	Entertainment
151	4	7	Engineer
596	3	7	Educator
443	3	36	Engineer
628	2	15	Student
323	3	28	Administrator
64	1	33	Programmer

Time taken:109.654 seconds, Fetched: 4556 row(s)

3.2.5. Simple View: View is a predefined named query stored in the database [19]. They are queried in the same way that tables can be queried. They contain the rows and columns like the tables do. However, views are not tables stored in; they are calculated from one or more tables. Views can join data from multiple base tables or other views and aggregation of function to summarize data.

```
hive>  create view current_movies_list as
       select distinct userid,timestamp
       from movies
       where rating > 2;
       OK
       Time taken: 0.746 seconds

hive>  select * from current_movies_list;
```

Userid	Timestamp
943	8.8863990388
943	8.8863990298
943	8.8863990538
943	8.8863990728
943	8.8863990278
943	8.8863990068
943	8.8863990678
943	8.8863990728
943	8.8864012588
943	8.8864014388
943	8.8864018688

Time taken: 121.992 seconds, Fetched: 435 rows

3.2.6. Lateral View: Lateral View is used with user defined table generating functions (UDTF), a UDTF generates 0 or more output rows for each input rows .A Lateral View first applies UDTF to each row of base table and then joins each output rows to input rows [20].

UDTF function was introduced in Hive 0.7

```
hive>  select * from pageAds
       front_page                [1,2,3]
       Contact_page              [3,4,5]

hive>  select pageid, adid_list
       from pageAds LATERAL VIEW explode(adid_list)
       adTable AS adid;
```

pageid(string)	adid(int)
“front_page”	1
“front_page”	2
“front_page”	3
“contact_page”	3
“contact_page”	4
“contact_page”	4

3.3 Results

We have examined the execution time of the queries executed on the Hive with the same queries in RDBMS and we found that response time of the queries in Hive is much better than that of RDBMS for Large data sets.

3.4 Conclusion and Future Work

We have executed and reviewed various queries implementation on Hive for large datasets by using different query implementation techniques like Indexing, Column Statistics, Lateral View, Join operations etc. Mapping and reducing functionalities of Map-reduce and HDFS helped Hive to process larger and unstructured datasets. Traditional database system like Oracle, MySQL, and PostgreSQL is highly optimized as compared to Hive. Hive is less optimized so optimization in Hive has good research possibilities. Also, Hive does not support Update and Delete functionality yet. So research can be progress in this area.

4. ACKNOWLEDGEMENTS

We are thankful to Mody University of Science and Technology, Lakshmangarh, Rajasthan, India for the valuable support in our research.

REFERENCES

[1] “Gartner IT Glossary,” http://www.gartner.com/it-glossary/big-data/, 2013.

[2] “The 7 V’s of Big Data,” http://mbitm.uts.edu.au/ feed/7-vs-big-data, 2007.

[3] “What is Apache Hadoop,” http://hortonworks. com /hadoop/,2011-2014

[4] “Fern Helper, Bringing big data to the enterprise ,” http://www 01.ibm.com/software/ in/data/ bigdata/, January 2012

[5] “Apache Hive,” http://hortonworks.com / hadoop /hive/,2011-2014

[6] Anja Gruenheid, Edward Omiecinski, Leo Mark “Query optimization using column statistics in hive,” Proceedings of the 15th Symposium on International Database Engineering & Applications, September 2011, pp. 97-105

[7] “Downloads,” https://hive.apache.org /downloads. html,2011-2014

[8] “GroupLens,” http://grouplens.org/datasets /movielens/,2015

[9] Apichon Witayangkurn, Teerayut Horanont, Ryosuke Shibasaki “Performance comparisons of spatial data processing techniques for a large scale mobile phone dataset, ” Proceedings of the 3rd International Conference on Computing for Geospatial Research and Applications ,Article no.25 ACM New York, NY, USA , June 2012,pp. 1-6

[10] Zhang,S.,et. al, “Spatial queries evaluation with Mapreduce,” In Proceedings of International Conference on Grid and Cooperative Computing 2009,IEEE Computer Society, 2009, pp 287-292

[11] Zhang,S.,et.al, “SJMR:Parallelizing spatial join with Mapreduce on Clusters,” CLUSTER, 2009, pp 1-8

[12] Huai Yin,Chauhan Ashutosh,Gates Alan,Owen “Major technical advancements in apache Hive” Proceedings of the ACM SIGMOD international conference on Management of data ,June 2014, pp.1235-1246

[13] Floratou Avrilia,Teletia Nikhil,Zhang Donghui "Can the elephants handle the Nosql onslaught?" Proceedings of the VLDB Endowment,Volume 5 Issue12,August 2014,pp.1712-1723

[14] Thusoo Ashish, Shao Zheng,Murthy Raghotham "Data warehousing and analytics infrastructure at facebook " Proceedings of the 2010 ACM SIGMOD International Conference on Management of data ,June 2014,pp. 1013-1020

[15] "The Apache Software Foundation ," http: //docs. hortonworks.com/HDPDocuments/HDP2/HDP-2.0.0.2/ds_Hive/authorization.html,1999-2013

[16] " Truncate command(Hive vs. SQL)," http:// hadoop.drawbackz.com/stack/12075/truncate-command-hive-vs-sql.html,

[17] "Hive Table Statistics,"http://www.cloudera. com/ content/cloudera/en/documentation/core/latest/topics/cm_mc_hive_table_stats.html,2015

[18] http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.0.0.2/ds_Hive/optimize-joins.html# JoinOptimization-Optimize Chainsof MapJoins, 1999-2013

[19] "HiveQL Data Definition," https://www. safari booksonline.com/library/view/programming-hive/9781449326944/ch04.html,2013

[20] "Hive Lateral View," http://hive_javadocs/ql/org /apache/hadoop/hive/ql/exec/LateralViewJoinOperator.html,2013

About the Author

Ms. Barkha Jain has completed her B-Tech from RTU, Kota in the year 2009-2013.She is currently pursuing M-Tech from Mody University, Lakshmangarh. She has presented various technical papers at Intra and Inter college levels. Her research areas include Big Data Analytics and Database Systems.

Manish Kumar Kakhani received his Bachelors and Masters in Technology in Information Technology from ABV-Indian Institute of Information Technology and Management, Gwalior, India in 2010.In 2010; he joined as a faculty of at KIIT University, Bhubaneswar, India where he worked for two years as an assistant professor in School of Computer Science and engineering. Currently, He is a Faculty of College of Engineering and Technology at Mody University, Lakshmangarh, India since 2012. He has organized ISTE Workshop on computer Programming under National Mission on Education through ICT (Ministry of Human Resource and development, Govt. of India).His research interests are in the areas of Big Data analytics and information security. He has been supervising two M.Tech. students.