IJIRR

International Journal of Information Research and Review
Vol. 05, Issue, 05, pp.5463-5466, May, 2018

REVIEW ARTICLE

DATA STORAGE MECHANISMS IN HIVE

***Subash Thota**

Data Architect, Crgt, USA

ARTICLE INFO

Article History:

Received 19th February, 2018
Received in revised form
29th March, 2018
Accepted 20th April, 2018
Published online 30th May, 2018

Keywords:

Big Data Analytics, Social Analytics, Storage Analytics, Data Management, Information Quality, Data Mitigation, Metadata, Data Profiling

ABSTRACT

The journey of Hadoop and its eco-system is getting more and more efficient to serve better in the Big Data arena. In this paper, we will explore and experiment the technical capabilities of the latest Apache Hive version 0.12. One of the key features that attract usto Hive is the data storage mechanism. In the present situation, we are undergoing a revolution in the way that data is collected. From the evolution of Big Data, we started sailing deeper into the massive set of complex data, either structured or unstructured, to perform Business Intelligence and Predictive Analysis. The data storage mechanism plays akey role in writing, reading and storing the data, effectively.

INTRODUCTION

Big Data awareness and its growing significance to business have been reaching fast to a greater range of audience. The market for data is exploding:

- 2.7 zeta bytes of data exist in the digital universe today.
- The Obama administration is investing $200 million in Big Data research projects.
- IDC estimates that by 2020, business transactions on the Internet, both for Business-to-Business (B2B) and business-to-consumer (B2C), will reach 450 billion per day.
- Facebook stores, accesses and analyzes more than 30 petabytes of user-generated data.
- More than 5 billion people are calling, texting, tweeting and browsing on mobile phones worldwide.
- Decoding the human genome originally took ten years to process; now it can be achieved in one week.

To put this in perspective, the production of data is expanding now at an astonishing pace. Experts point to a 4300 percent increase in annual data generation by 2020.Big Data solutions consume volumes of data that are enormous and help detect very complex patterns that are very difficult to see, without massive data stores being analyzed. One of the technologies most often associated with the era of Big Data is Apache Hadoop, and Apache Hive is the de facto standard for SQL in Hadoop with more enterprises relying on this open source roject rather than any other alternative.

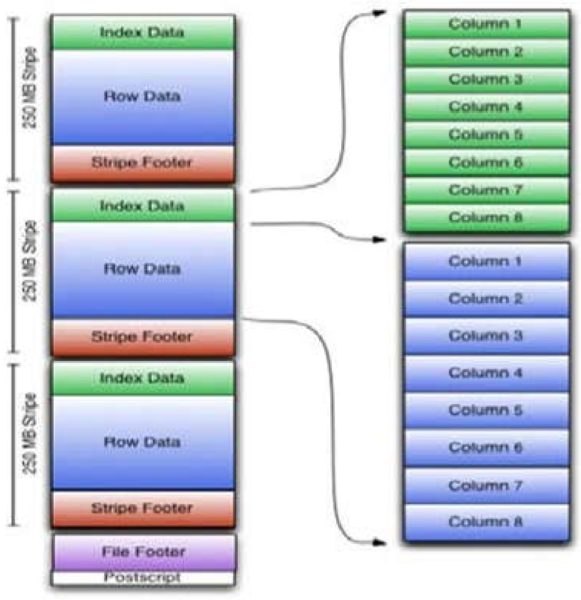
****Corresponding author:*** **Subash Thota,**
Data Architect, Crgt, USA.

Although there is much technical information on Hadoop, there is little about how to effectively structure and store data in a Hadoop environment. As opposed to relational data modeling, structuring data in the Hadoop Distributed File System (HDFS) is a relatively new and a constantly evolving domain. Even though the intrinsic nature of parallel processing and the Map-reduce algorithm provide an optimal environment for processing Big Data quickly, the structure of the data itself plays a vital role. In this paper, we explore the techniques used for data modeling in Hadoop environment. Specifically, the experiments described in this paper intend to determine the best structure and physical modeling technique for storing data in a Hadoop cluster using Apache Hive 12,to enable efficient data access.

Although other software also interacts with Hadoop, our experiments are focused on Apache Hive. Apache Hiveis an open source data ware housing solution builton top of Hadoop that manages and queries structured data. I tutilizes Map-reduce for execution, an RDBMS for its system catalog (called Hive Meta Store) and HDFS for datastorage. We are mainly interested in the effective storage of data in HDFS, using Hive, and the storage file format plays a key role in it. Hive 0.10 supports file formats for storing data -Text File, Sequence File, RC File and Parquet File, till its release 0.10. But the introduction ORC file format from Hive 0.11 made a big change in data storage and retrieval performance. Therefore, we continue to explore the new features of Apache Hive 0.12 which is the latest version and run some experiments to interpret the results. In this paper, we concentrate on ORC File-format, and in future papers, we try to cover the other features of Hive 0.12 such as Hive Query Optimizer, Predicate Push Down, Vectorized Query execution, and so on.

ORC FILE Format

This diagram illustrates the ORC file structure



- An ORC file contains groups of row data called stripes
- The default stripe size is 250 MB.
- Each stripe consists of - index data, row data, and stripe footer
- Row data is used in table scans.
- Stripe footer contains a directory of stream locations
- Large stripe sizes enable large, efficient reads from HDFS. File footer –holds auxiliary information.

Postscript

- Compression parameters
- Size of the compressed footer

An ORC file contains groups of row data called stripes, along with auxiliary information in a file footer. At the end of the file, a postscript holds compression parameters and the size of the compressed footer. The default stripe size is 250 MB. Large stripe sizes enable large, efficient reads from HDFS. The file footer contains a list of stripes in the file, the number of rows per stripe, and each column's data type. It also contains column-level aggregates count, min, max, and sum.Compared to RC File format, for example, ORC file format has many advantages such as:

- Asing lefileas the output of each task, which reduces the Name Node'sload.
- Hive type support including date time, decimal, and the
- Complex types (struct, list, map, andunion).
- Light -weight indexes stored within the file Skip row groups that don't pass predicate filtering Seek to a given row.
- Block-mode compression based on data type run-length encoding for integer columns dictionary encoding for string columns.
- Con current reads of the same file using separate Record Readers.
- Ability to split files without scanning for markers.
- Bound the amount of memory needed for reading or writing.
- Metadata stored using Protocol Buffers, which allows addition and removal of fields.

Data Environment

Our main focus is to experiment the Hive storage format and measure the performance of data compression and data manipulation process.

Data Storage Experiments

To test the various data storage formats in Hive, we have identified typical scenarios to measure the performance of Data storage capabilities and the performance of Data manipulation process. Following metrics will be used to measure the performance:

- The data size in HDFS for different storage file-formats on a table without using compression codec.
- The data size in HDFS for different storage file-formats on a table using compression codec.
- From the above task measure the time taken to write the Data into HDFS.
- Perform typical Group by operation to identify the peak selling day across all the stores and list out the top 10 stores which sold this item.
- Drills down further more to identify the Top 10 selling stores by revenue.

All these experiments are performed in HDP 2.0 environment; a Horton works distribution of Hadoop and Apache Hive version 0.12 was used. All the queries we ran in Hive prompt and all the HQL scripts have been run thrice on a quite environment to obtain the accurate performance information. We use Snappy Compression Codec. Wherever compression is required.

EXPERIMENT RESULTS

Interpretation of Results

Experiment 1: Data Compression

From the results, we are able to see the capability of ORC file format on data compression. The ORC file format helps to compress data up to the extent of 90% from the actual size. While comparing ORC compression ratio with RC file-format, the data was compressed 10 times more than RC-file. The result of ORC file format was even better when applying the data compression codec. Also in our experiment, we have noticed that ORC file-format works very well on huge set of data instead on a small volume of data. The result clearly predicts the ability of ORC file format on data compression.

Experiment 2: Data Writing Speed

We have tested this experiment on a retail real time data set of data volume more than 11GB and we able to see that there was a better write performance when compared with text file-format and even with RC file-Format.

Without Applying Compression Codec

HDFS Storage Data Size	File Format	Flat File Size	HDFS Storage Size	Size Ratio Actual Vs HDFS	Size Compressed %
	TEXT FILE	11.5 GB	11.38 GB	1.01	1.04
	RC FILE	11.5 GB	1.2 GB	9.58	89.59
	ORC FILE	11.5 GB	0.13 GB	88.46	98.87

Applying Compression Codec

HDFS Storage Data Size	File Format	Flat File Size	HDFS Storage Size	Size Ratio Actual Vs HDFS	Size Compressed %
	TEXT FILE	11.5 GB	12.15 GB	0.92	-5.6
	RC FILE	11.5 GB	0.16 GB	71.87	98.6
	ORC FILE	11.5 GB	0.07 GB	164.28	99.4

Experiment 2: Data writing speed

Without Applying Compression Codec

Writing Speed	File Format	Min. Time (Sec)	Max. Time (Sec)	Average (Sec)
	TEXT FILE	1796.73	2585.65	2241.32
	RC FILE	684.29	994.27	825.72
	ORC FILE	744.51	751.28	747.895

Applying Compression Codec

Writing Speed	File Format	Min. Time (Sec)	Max. Time (Sec)	Average (Sec)
	TEXT FILE	1225.751	1698.11	1461.93
	RC FILE	1028.134	1602.28	1315.207
	ORC FILE	997.95	1241.74	1119.845

Experiment 3: Data Reading Speed

Query1: Perform typical Group by operation and JOIN conditions to identify the visit date which made the maximum revenue across all the stores and list out the top10 stores

Reading Speed	File Format	Min. Time (Sec)	Max. Time (Sec)	Average (Sec)
	TEXT FILE	295.208	299.35	297.279
	RC FILE	259.669	263.451	261.605
	ORC FILE	90.47	110.318	100.394

Query 2: Drills down further more to identify the top 10 selling stores by revenue

Reading Speed	File Format	Min. Time (Sec)	Max. Time Sec)	Average (Sec)
	TEXT FILE	166.362	172.424	169.393
	RC FILE	102.294	106.59	104.442
	ORC FILE	40.312	43.527	41.919

From the stats we are able to see 300% faster data ingestion when compared to Text file-format and the writing speed is comparable to RC file-format. In the best case scenario, it is up to 10% faster than RC file-format.

Experiment 3: Data Retrieval Speed

To check the performance of data retrieval we tried with few business queries. To capture the ideal result, we have executed the queries thrice.

Query1

The query in volves GROUP BY operations, In-line view and JOIN condition store trieve the peak selling date across all stores. From the stats it was very clear that ORC file-format out performed when compared with other file-formats. The ORC file-format was 3times faster than others file formats and this result was achieved without applying any performance tuning methods.

Query2

This is very simple query which has GROUP BY operation and In-line view. The result was slightly different when compared with query 1and time taken to retrieve the data was little higher. But the ORC file-Format maintains its data retrieval performance when compared with other file formats. But we are able to find some difference in ORC file-format data retrieval pattern when we join tables which were created using different file-format. The time taken was almost equal to that as the RC-file format.

Conclusion

We are into a new arena with massive data sets in different formats and the data challenge has just got up a smarter notch in the industry. The Big Data solutions simply consume volumes of data that are enormous in size, and help to detect very complex patterns that are very difficult to see, without massive data stores being analyzed. Through our experiments, we have shown that the Hive file-format plays a key role in storing and retrieving the data efficiently, enabling Hive to support interactive workloads and preserve investments. The Hive ORC file-format is definitely a game changer for all real-time clusters and in the case of clusters which are either maxed out or close to max out this is the only viable option. Future experiments should consider performance enhancement provided by Hive 0.12 such as ORC File and Vectorization, ORC Predicate Push down, Reading ORC file from Map Reduce using Hcat, and so on. As Big Data technology continues to advance, the features that are available for structuring data will continue to improve, and further options for improving data structures will become available.

REFERENCES

Big Data Analytics with R and Hadoop Vignesh Prajapati, Packt Publishing, 1st edition, 2013.

Hadoop Beginner's Guide Garry Turkington, Packt Publishing, 2013.

Hadoop For Dummies Dirk deRoos, Paul C. Zikopoulos, Bruce Brown, Rafael Coss, and Roman B. Melnyk, John Wiley & Sons, Inc., 1st edition 2014.

hortonworks.com/blog/announcing-apache-hive-0-12.

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC.

Machine Learning with R Brett Lantz, Packt Publishing, 1st edition, October 2013.

Thota, S. 2017. Big Data Quality. Encyclopedia of Big Data, pp.1-5. https://link.springer.com/referenceworkentry/10.1007/978-3-319-32001-4_240-1.

****