A Comparison of HDFS File Formats: Avro, Parquet and ORC

Article *in* International Journal of Advanced Science and Technology · June 2020

DOI: 10.5281/zenodo.7027910

Supreeth Shivashankar
Reva University
17 PUBLICATIONS **49** CITATIONS

A Comparison of HDFS File Formats: Avro, Parquet and ORC

Ashok Shivayogappa[1], Supreeth S[2]

[1]*Student, School of Computing and Information Technology, REVA University*

[2]*Assistant Professor, School of Computing and Information Technology, REVA University*

Bengaluru, India

[1]*mamani.asm@gmail.com*, [2]*supreeth1588@gmail.com*

Abstract

Hadoop is one of the standard platforms for managing and storing Big Data in distributed systems. But the lack of good developers to write Map Reduce programs into the development environment has pushed the adoption of SQL based query system into the Hadoop Ecosystem in an attempt to benefit from the traditional relational database systems in terms of the skills, especially in the Business process and Intelligent Analytical process. On top of the Hadoop environment a new framework has arrived ie; Apache Hive as the standard data warehouse engine. This is one of the challenges by the industry-leading developers to work on improvements both in query execution and as well as in data storage paradigms.

In this work, various data structure file formats like Avro, Parquet, and ORC are differentiated with text file formats to evaluate the storage optimization, the performance of the database queries. Various kinds of query patterns have been evaluated. The output of the study shows that ORC and Parquet file format takes up less storage space compared with Avro and text files format, it is because of binary data formats and compression techniques used. Furthermore, Aggregate queries of the ORC and Parquet data structures are quicker compared with Avro or text file formats because earlier two formats support well for column-based queries.

Keywords: *Hadoop, HIVE, Avro, ORC, Parquet*

1. Introduction

The broad utilization of online life, web of things and different web/portable applications are catching gigantic measure of information. Every single second people leave a meaning full usable amount of data behind them in this digital world. Hence in today's world data is called fuel for life. If this data is used carefully companies can get business advantages such as generating product recommendations, inventing new products, analyzing the market, etc. and even data analysis can provide few early key indications that can turn the fortune of business and provide rooms for further more analysis. Every single second generates a huge amount of raw data, without proper processing, no one can get benefits out of it. Thus for the better investigation yield handling of this crude information now and again end in a database, which permits further to process and examine in various manners[1].

As I mentioned above data is processed in the database, RDBMS is not suitable for storing and processing a large amount of data, called “Big Data” which is in the form of large files, images, and video. And not only large, but raw data may also be a structured, unstructured, or semi-structured one. Hence RDBMS, not a good choice when it comes to advanced analytics. To handle massive quantities of different categories of data we need to have a distributed and real-time processing system, which is Hadoop[2]. Hadoop is an open-source project of Apache foundations. It is a framework used for processing big data, now a day's Hadoop[7] is a core part of the computing infrastructure for companies such as Yahoo, Google, Facebook, LinkedIn, Twitter, etc.

There are many sources for big data like sensor activity records from electronic devices, social interactions like Facebook and Twitter, OLTP business transactions, electronic files, and broadcastings.

Generally, crude information is put away in singular content arrangements, model CSV, JSON, XML. These arrangements of records are in the comprehensible organization and are organized and simple to peruse and alter. But, storing raw data as a plain text format has some disadvantageous like more storage space is required and even bigger problem is with Hadoop because in Hadoop ecosystem files data is stored in data blocks with some replication factor for each data blocks. To get complete advantages of Hadoop[8], we must be able to split file data into the same size as Hadoop storage block sizes and compress them independently (using snappy or LZO compression techniques). If file format does not support splitting and compression, then this could generate a substantial performance penalty.

The type of file formats is like Avro, Parquet, RC and ORC are of binary data storage formats these kinds of formats are also used in Hadoop for storing and processing data. Since these are of binary formats, they support splitting and block compression and enjoy broad, relatively mature, tools support in the Hadoop ecosystem.

2. Literature Survey

Hadoop technology has become the heart of all big data projects. Hadoop has gotten standard for putting away and handling enormous information for unstructured, yet also for organized information as well. HDFS is used for storing a large set of data and acts as a high-speed data source for many applications. As a result, having SQL analysis functionality for HDFS has become most important. There are various tools available for this, but HIVE stands on top of all, which is supports SQL like analysis to data stored on the HDFS cluster[3] and the same HIVE has been chosen for this comparison study. Figure 1 represents File format Structures (a) Avro File Structure (b) Parquet File Structure (c) ORC File Structure.

2.1 Hive:

Apache Hive is a foundation for information distribution centers based on Hadoop. The Hive was initially evolved by Facebook and supports the analysis of big informational indexes put on HDFS by querying upon SQL-like decisive question language, called HiveQL. It doesn't carefully adhere to the SQL-92 standards. Moreover, locally calling User Defined Functions (UDF) in HiveQL permits sifting information by custom Java or Python scripts. Connecting custom contents in HiveQL makes the usage of locally unsupported explanations conceivable. Hive comprises of two center segments: the driver and the Metastore. The driver is liable for tolerating HiveQL articulations, submitted through the order line interface (CLI) or the HiveServer, and interpreting them into occupations that are submitted to the MapReduce[6] engine. This permits clients to break down enormous informational indexes without really creating MapReduce programs themselves. The Metastore is the focal storehouse for Hive's metadata and stores all data about the accessible databases, tables, table sections, segment information types, and the sky is the limit from there. The Metastore utilizes normally a customary RDBMS like MySQL to persevere the metadata. Table 1 shows the comparison of different HDFS file formats.

Table 1: Comparison of different HDFS file formats

	CSV	**AVRO**	**ORC**	**PARQUET**
Storage Orientation	Row	Row	Column	Column
Compression	All	Snappy and Deflate	ZLib and Snappy	Glib, LZO, and Snappy
Schema Storage	No	Yes	Yes	Yes
Schema Evaluation	No	Yes	Limited	Limited
Complex dataType	No	Yes	Yes	Yes
Random access	No	No	Yes	Yes
Data Append	Yes	Yes	Yes	No

2.2 Avro:

Avro is a language-neutral, a schema-based serialization technique. And it is a preferred tool in Hadoop for data serialization. Each read and write operations associates with a language-independent schema. It serializes data into a compact binary structured format, which later will be deserialized by data consuming application. Avro makes use of JSON format for schema structure store, this format is supported by many languages like C, C++, Python, Ruby, etc. Binary Structured format created by the Avro tool can be compressible and splittable. Hence Avro is preferred in Hadoop MapReduce jobs.

2.3 Parquet:

The main idea of creating a Parquet file format is to get the advantage of compressed, efficient columnar data representation on HDFS/Hadoop projects. It is built on the base of complex nested data structures, which makes use of record shredding and assembly algorithm as described [4]. Parquet is a column-based storage format, where the values of the same data type are stored together, this enables better compression. This method of organizing data in columnar fashion is good for the query which reads specific columns by minimizing IO. In the case of big data projects as the number of data increases, the cost of storage and processing also increases. Parquet is a good choice of bigdata as its server both needs efficiency and performance in both processing and storage.

2.4 Optimized Row Columnar:

Optimized Row Columnar (ORC)[5] is a good way to store Hive data. Using the ORC file format improves the performance when Hive writing, reading, and processing of the data. ORC has many advantages like it supports hive data type including complex data types, it has light-weighted index stored inside file - this helps to skip a group of rows that do not pass filter conditions. ORC also supports block-mode compression based on datatype; the ORC file can split files without scanning for the marker. Another advantage of ORC is that metadata is stored using a protocol that allows the addition and removal of fields.

ORC file structure:

An ORC file contains the groups of rows data called Strips along with auxiliary information in the footer of the file. At the end of this file, postscript holds the compression parameter and size of the compressed version of the footer.

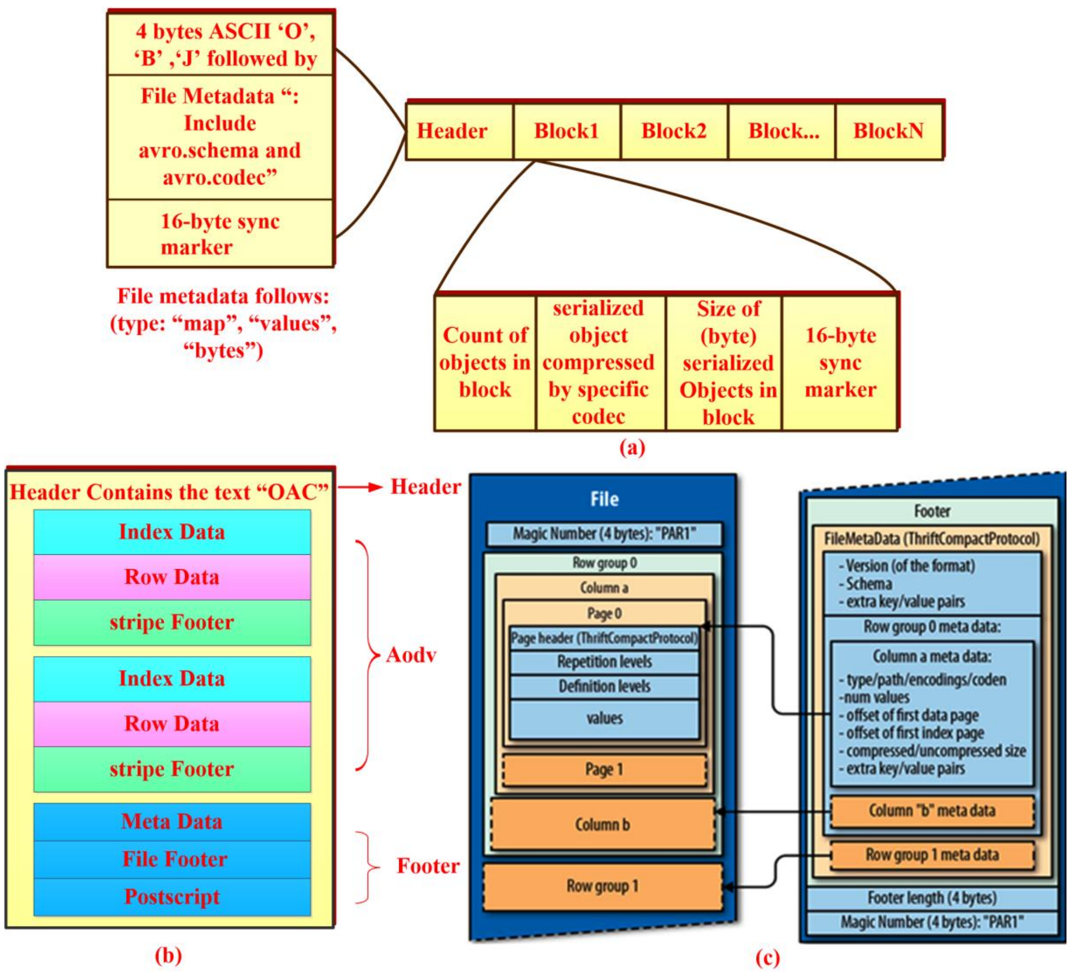
4 bytes ASCII 'O', 'B' ,'J' followed by
File Metadata ": Include avro.schema and avro.codec"
16-byte sync marker
File metadata follows: (type: "map", "values", "bytes")
Header
Block1
Block2
Block...
BlockN
Count of objects in block
serialized object compressed by specific codec
Size of (byte) serialized Objects in block
16-byte sync marker
(a)
Header Contains the text "OAC"
Index Data
Row Data
stripe Footer
Index Data
Row Data
stripe Footer
Meta Data
File Footer
Postscript
Header
Aodv
Footer
(b)
File
Magic Number (4 bytes): "PAR1"
Row group 0
Column a
Page 0
Page header (ThriftCompactProtocol)
Repetition levels
Definition levels
values
Page 1
Column b
Row group 1
Footer
FileMetaData (ThriftCompactProtocol)
- Version (of the format)
- Schema
- extra key/value pairs
Row group 0 meta data:
Column a meta data:
- type/path/encodings/coden
-num values
- offset of first data page
- offset of first index page
- compressed/uncompressed size
- extra key/value pairs
Column "b" meta data
Row group 1 meta data
Footer length (4 bytes)
Magic Number (4 bytes): "PAR1"
(c)

Figure 1: File format Structures (a) Avro File Structure (b) Perquet File Structure (c) ORC File Structure

3. The proposed work

This study aims to compare the Hadoop file formats such as Avro, Parquet, and ORC in terms of storage space required on HDFS, query performance especially aggregate queries, which are a major part of data analysis and compression technique which improves the speed and efficiency of data processing with Hadoop tools, like HIVE.

At the end of this proposed work there must be clear answers to the following questions:

RQ1: Which file format consumes less storage space of HDFS?

RQ2: Which data structure format (Avro, Parquet, or ORC) supports high performance with regards to aggregated/scanned queries?

RQ3: Which data format (Avro or Parquet or ORC) is more compact?

To get answers to the above questions, the experimental investigation has been selected as a research method. The research method has five stages: it starts with planning, operation, analysis, interpretation, and report.

Avro, Parquet, and ORC file format are selected for this experiment were based on assumption that row-based access supported by Avro provides better performance on scan queries, where all columns are used for processing. In contrary Parquet and ORC, both are columnar file formats, provide the best performance on column-oriented queries i.e. specific columns are scanned and selected for processing. Following are some more assumption made to experiment the problem (are called null hypothesis)

- H_0^A Data Structured format Avro is a good choice for **scan-based queries**
- H_0^B Data structured format Parquet and ORC are a good choice for **aggregation queries**
- H_0^O Between Parquet and ORC formats, ORC is a better choice for both scan and aggregation queries.
- H_0^C **Compactness** consumed by ORC, in compared with the other two is less

Each hypothesis H_0^Z where Z refers to a certain quantity (A for Scan-based queries, B for aggregation queries, and C for storage space) has been measured by corresponding random variable A^Z, $P^{Z,}$ and O^Z respectively Avro, Parquet, and ORC file formats. For example, H_0^C tests the storage space of the data format Avro A^C, $P^{C,}$ and O^C. Therefore, the null hypothesis H_0^C is expressible as:

$$H_0^C: p\,(A^C > P^C > O^C) = p\,(O^C > P^C > A^C) \qquad (1)$$

That is, the probability p that Avro file storage space is more compare to Parquet file storage space and Parquet file storage space is more when compared with ORC file storage space. Correspondingly, the alternatively hypothesis H_1^C is that there is difference in probability:

$$H_1^C: p\,(A^C > P^C > O^C) \neq p(O^C > P^C > A^C) \qquad (2)$$

Cluster Configuration:

Today there are many different big data management solutions, like Hortonwork's HDP, Cloudera's CDH, Apache Spark, Microsoft's Dryad Oracle's Big Data Appliance, Apache Hadoop, etc. Every one of these frameworks for the most part centered on huge information stockpiling and handling; be that as it may, they may contrast in approaches. For example, MapReduce preparation varies from Spark's DAG approach. In the present work, the Hortonworks Data Platform. The Hadoop version on the cluster is 2.7. The main reason for that is the high popularity of the platform because of its open to the developers. Hortonworks has consolidated more open-source Hadoop biological system's ventures than some other stage. Consequently, it prompts greater notoriety among endeavors since it doesn't prompt seller lock-in.

For the test examination, a 5 hubs bunch has been picked, arranged, and intended for large content organization information preparing. There is one node is used as a name node. The remaining 4 data nodes run the worker roles for the Hadoop services.

Data nodes in the clusters have Intel® processor 2.5GHz, 500GB HDD with 8 cores and 32 GB RAM.

Data Model and Sample Data:

In today's internet world there exists different datasets and raw data for experimenting. However, for this study I have created own data sets using an online data generator tool – using this we can generate up to 100 combinations of data format and information. This tool also provides facilities for exporting generated sample records to CSV, JSON, and Excel file formats.

Using this online data generator created 10 (ten) sample data columns and generated 1M sample records. Below Table 2 shows the column and their data types.

Table 2: PartTable Columns

Column Name	Date Type
ObjID	INT
PartNumber	String
mfrID	String
Category	String
Country	String
Status	String
TreeID	INT
Part Desc	String
BOM Number	INT
CategoryName	String
LCInfo	String

All records are exported into .csv format (with the file name as partTable.csv) and .csv file has been uploaded into HDFS as a plain text file. HDFS "put" command is used for loading partTable.csv file HDFS. Command to move a file from local to HDFS is shown below.

```
hdfs dfs -put parttable.csv /user/mamaniasm7954/
```

Move Data to Hive Table:

For loading data onto Hive tables. Hive managed tables are created using CREATE TABLE statement with "STORED AS AVRO", STORED AS PARQUET" and STORED as ORC" option, one for each file format, below queries shows CREATE table statement for all 3 tables using HIVE scripts.

```
create table sampleData_avro
(objid int,      partnumber STRING, mfrid STRING,
category STRING, country STRING,    status int,
treeid int,      partDesc STRING,   bomnumber int,
categoryname STRING,lcinfo STRING)
row format delimited
fields terminated by ','
```

```
stored as AVRO;

create table sampleData_parquet
(objid int,     partnumber STRING, mfrid STRING,
category STRING, country STRING,   status int,
treeid int,     partDesc STRING,   bomnumber int,
categoryname STRING,lcinfo STRING)
row format delimited
fields terminated by ','
stored as PARQUET ;
TBLPROPERTIES ("orc.compress"="SNAPPY");

create table sampleData_orc(objid int,     partnumber STRING,
mfrid STRING,
category STRING, country STRING,   status int,
treeid int,     partDesc STRING,   bomnumber int,
categoryname STRING,lcinfo STRING)
row format delimited
fields terminated by ','
stored as ORC;
```

An intermediate hive table, called sampleData_text, is created to load data from .csv file into Hive text file and data has been loaded onto the same using LOAD command as shown below and Post that data is converted into Avro, Parquet and ORC formats using HIVE DML statements are as shown below.

```
LOAD DATA LOCAL INPATH '/home/mamaniasm7954/merge_from_ofoct.csv' INTO
TABLE sampleData_text;


INSERT OVERWRITE TABLE sampleData_avro select * from sampleData_text;
INSERT OVERWRITE TABLE sampleData_parquet select * from sampleData_text;
INSERT OVERWRITE TABLE sampleData_orc select * from sampleData_text;
```

Storage size on HDFS for a different file format is given in table 3 and figure 2.

Table 3: Table Name and record count, its record count, its storage size of HDFS in different file formats.

		Size in MB			
Table Name	Record Count	Text File	.Avro File	Parquet File	ORC File
PartTable	1,000,000	140	135	88	48
Manufacture Table	12,000	36	34	20	8
Category Table	4430	13	12	7	4

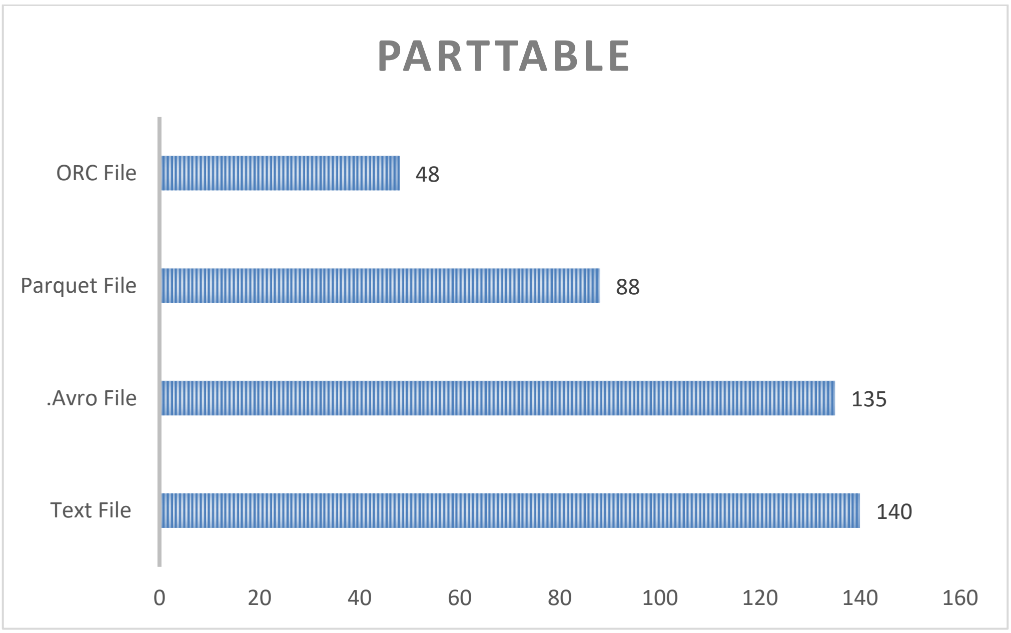

Figure 2: Storage Size of different file formats

Queries:

Various queries have been constructed and executed on the sample data set. All queries are classified into aggregation and scan queries. The execution time of each query on different file structure is observed and listed in table 4 below.

Query1 and Query2 used to select data from Avro, Parquet, and ORC data files query is as follows:

Query 1:

```
select category, count(*) from sampledata_parquet group by category; --21.87---24 seconds 750 msec
select category, count(*) from sampledata_avro group by category;--54.65
select category, count(*) from sampledata_orc group by category; -- 22.34s
```

Query 2:

```
select a.category,a.bomnumber, a.TopBoms
from (select category, bomnumber, rank() over(partition by category  order by bomnumber desc) TopBoms
    from sampleData_avro) a
where a.TopBoms < 6

select a.category,a.bomnumber, a.TopBoms
from (select category, bomnumber, rank() over(partition by category  order by bomnumber desc) TopBoms
    from sampleData_parquet) a
where a.TopBoms < 6

select a.category,a.bomnumber, a.TopBoms
from (select category, bomnumber, rank() over(partition by category  order by bomnumber desc) TopBoms
    from sampledata_orc) a
where a.TopBoms < 6
```

4. Results and Discussion

Sample data sets are loaded into Hadoop as the text file format and later converted to Avro, Parquet, and ORC format. This conversion, as shown in Table 3, shows a significant storage space economy. Figure 2 shows that ORC format takes 2.5 times less space compared to Avro and half of the space consumed by the Parquet file format. This is an answer to the first research question RQ1. Related to RQ3 with null hypothesis H_0^C proves alternative hypothesis H_1^C that there is a difference in compactness between data formats Avro, Parquet and ORC i.e; probability **p** is:

$$H_1^C\text{: } p\ (A^C > P^C > O^{C)} \neq p(O^C > P^C > A^C) \tag{3}$$

To answer the second research question RQ2, queries have been grouped into two categories as per hypothesis H_o^A and H_0^B. i.e.; Q3 & Q2 are scan queries and remaining all are aggregate type category queries.

The response to the subsequent research question RQ2 has been acquired by building the tests and estimating the execution time of the seven inquiries utilizing Beeline shell. Direct route shell has revealed time is near the time announced by CDH asset chief for similar inquiries. During the investigation, seven inquiries have been executed for each table, put away separately as Avro, Parquet, and ORC. Table 4 shows the time of these queries for each file format.

Table 4: Query execution time on each file format

Query	**Aggregation or scan**	**Query execution time(s,ms)**		
Query	Aggregation or scan	Avro	Parquet	ORC
Q1	AGR	54.65	21.87	22.34
Q2	AGR	30.2	18.35	15.4
Q3	SCAN	38.12	22.41	18.56
Q4	SCAN	53.12	36.41	24.13
Q5	AGR	40.24	19.34	17.33
Q6	AGR	64.05	31.37	28.34
Q7	AGR	53.12	37	24.13

As shown in Figure 3, Avro does the worst performance when compared with Parquet and ORC on both categories of queries (Scan and Aggregation). Thus there is a wrong null hypothesis H_0^A that data format Avro is better than the other two formats to scan queries. Because of the format of the data Parquet and ORC perform better than of Avro on both kinds of queries. Thereby null hypothesis H_0^B is true.

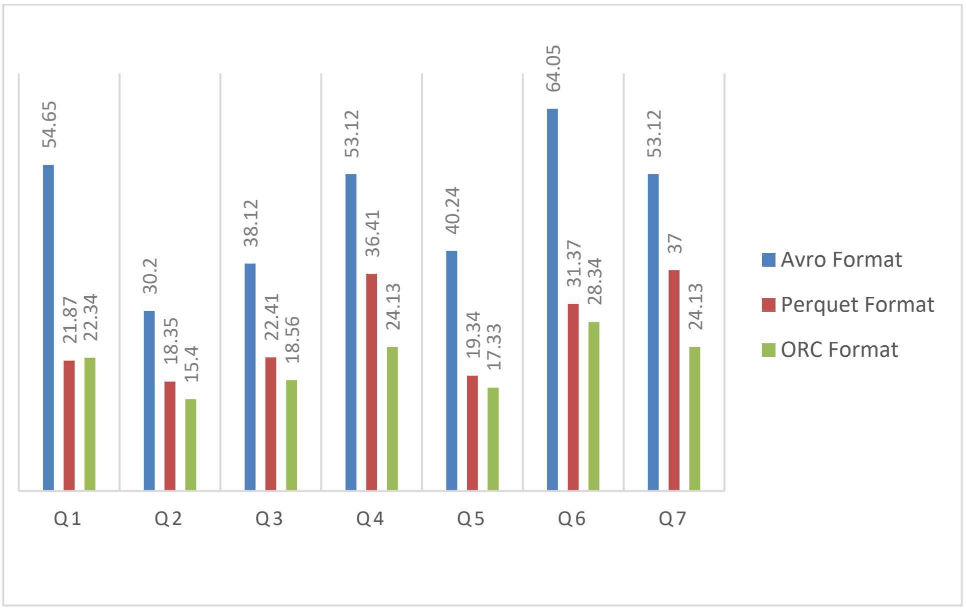

Figure 3: Execution time of each query on different formats

When we compare the execution time of Parquet and ORC, see Figure 4 below, ORC presents better performance than Parquet on both category queries, there for hypothesis, H_0^O holds good and are true.

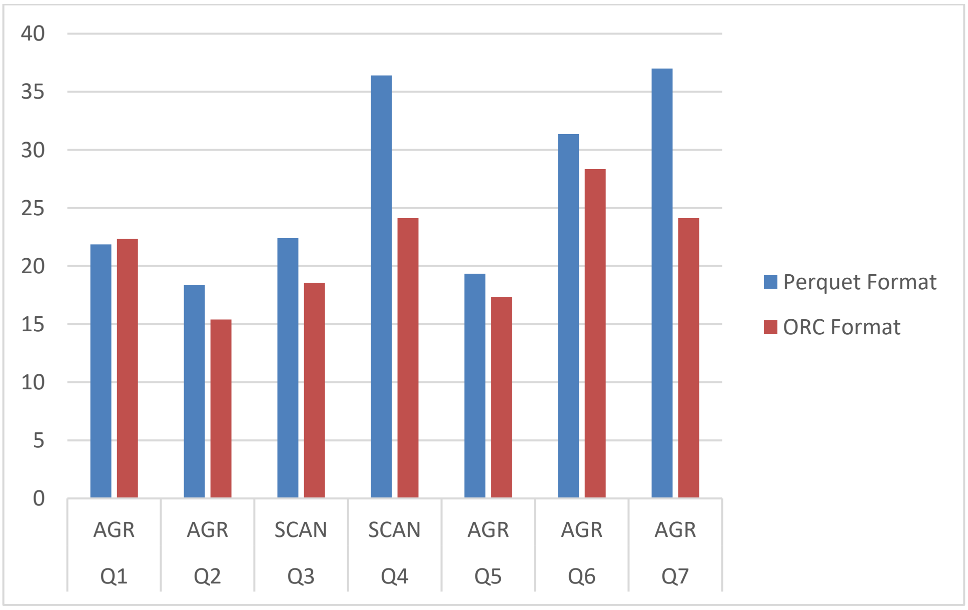

Figure 4: Execution time of each query on Parquet and ORC format

5. Conclusion

This work is performed within the scope of this research have been based on the comparison of HDFS compact data formats like Avro and Parquet. As a result of the comparison of HDFS compact data formats, literature work reviews a gap and that need for additional experiment and work have been formulated. In the recent big data ecosystem, ORC is emerging as a new file format and widely used across the companies for storing raw data in HDFS. Even though ORC is a column-oriented file format, like Parquet, it stands best compared with Parquet in case of compactness and query performances.

The current experiment and previous experiment show that Parquet format usage only worth using from the storage space economy point of view. Queries from the Parquet table are slow when compared with queries from ORC tables. Hence Parquet can provide 2 times faster execution time on average when compared with Avro. On the other side, ORC format gives 3 times faster execution on average rate when compared with Avro and 2 times faster execution on average compared with Parquet format.

Mostly at this movement, many companies are adopting big data fields for storing and analyzing their raw data. Thus a work conveying examination of record organizations and best practices for putting away information is the most required test. Even though a ton of investigation conveyed in scholarly work and the hole on the ORC design has been an inspiration for this exploration work.

References:

[1] Daiga Plase, Laila Niedrite, Romans Taranovs."A Comparison of HDFS Compact Data Formats: Avro Versus Parquet", Mokslas - Lietuvos ateitis, **(2017).**

[2] Jau-Ji Shen, Chin-Feng Lee, Kun-Liang Hou."Improved Productivity of Mosaic Image by Kmedoids and Feature Selection Mechanism on a Hadoop-Based Framework", 2016 International Conference on Networking and Network Applications (NaNA), **(2016).**

[3] Pramod Ravindra Patil, Vivek Kshirsagar. "Efficient time compression earthquake database using hadoop Hive ORC format", 2017 International Conference on Intelligent Computing and Control Systems (ICICCS),**2017.**

[4] Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, Theo Vassilakis, "Dremel: Interactive Analysis of Web-Scale Datasets Sergey", Proceedings of the VLDB Endowment, Volume 3, Issue 1, Sept 2010.

[5] Todor Ivanov, Matteo Pergolesi. "The impact of columnar file formats on SQL-on-hadoop engine performance: A study on ORC and Parquet", Concurrency and Computation: Practice and Experience, **2019.**

[6] Mohd RehanGhazi, DurgaprasadGangodkar, " Hadoop, MapReduce and HDFS: A Developers Perspective", Procedia Computer Science, Volume 48, **(2015)**, Pages 45-50

[7] Lulu, Qiu YanFeng, "An Optimal Solution of Storing and Processing Small Image Files on Hadoop", Procedia Computer Science, Volume 154, **(2019)**, Pages 581-587.

[8] Supreeth S., Raja Rajeshwari M.M, "Using Map-Reduce for Image Analysis in Cloud Environment", Proceedings of International Conference on Cognition and Recognition. Lecture Notes in Networks and Systems, vol 14, **(2018)**, Springer, Singapore.