# Performance Enhancement using Appropriate File Formats in Big Data Hadoop Ecosystem

**Vishal Naidu**

*Department of Electronics and Telecommunication, Ramrao Adik Institute of Technology, Mumbai, India*

---------------------------------------------------------------------***---------------------------------------------------------------------

**Abstract -** *Data drives the 21st century and many more generations to come. There are vast amounts of data being generated and collected every minute from various devices, software, emails, transactional data, organizational data, and user data collected by various big tech giants on the scale of zettabytes. As the data increases, the physical storage space also increases to cater to such humungous data. Storing them and preserving them becomes a significant challenge when data grows exponentially. The main challenge is storing this data and accessing it in real-time. This is where storage formats come into the picture. Different types of data formats are being used to store big data. This paper will compare five Data formats like Avro, orc, parquet, and textfile and their pros and cons and their usage.*

***Key Words***: **Big Data, Hive, Hadoop, ORC, Parquet, Avro, Performance Evaluation**

## 1.INTRODUCTION

The amount of data captured by enterprise organizations, social media, advertising companies, and various applications increases exponentially. Hadoop is one of the popular open-source Big Data frameworks in the industry today, capable of carrying out common Big Data related Tasks [3]. The file format that Hadoop supports is called the Hadoop distributed file system (HDFS). HDFS is a traditional way to store big data in Terabyte and petabyte-scale because of its reliability, scalability, and its fault-tolerant nature. HDFS contains the number of server machines/nodes on the scale of hundreds to thousands where the data is distributed and stored in parts. Traditional solutions are only efficient for specific file sizes or file formats [2]. HDFS provides high throughput access to application data and is suitable for applications that have large data sets [1]. The main advantage of HDFS is that it is highly fault-tolerant, and it can be deployed with low-cost commodity hardware. Handling and Understanding a Huge amount of data is a big challenge. HDFS supports many file formats but choosing a specific format depends on the use case, and many factors must be considered.

Raw data is generally stored in text format such as CSV, Text file, JSON, XML. These are also known as human-readable formats, i.e., they can be read and edited by humans. However, the issue with raw data formats is, it takes a tremendous amount of storage space. Storing Raw data in HDFS is again a more significant issue and in HDFS, each data is replicated three times. So, the amount of data taken by HDFS would be replicated three times, thus increasing the size by three times the original file size. Therefore, it becomes crucial to compress and store the files in HDFS. HDFS storage space utilization in a more efficient manner according to the task defined, and several binary data storage formats exist inside HDFS [4]. Some of them are RCFile, ORC, Avro, Parquet [4]. These formats are designed for systems that use MapReduce kinds of framework, and it is a structure that is a systematic combination of multiple components, including data storage format, data compression, and optimization techniques for data reading [4].

## 2. BACKGROUND

Big Data projects are frequently managed with Hadoop Technology. Hadoop is currently the basic standard and is used for analyzing large amounts of unstructured data, as well as certain structured data [7]. The Hadoop Distributed File System (HDFS) is intended to store very large data sets consistently and to transmit such data sets to user applications at high bandwidth [8]. As a result, offering SQL analytic capabilities to huge data stored in HDFS is becoming increasingly critical. Although alternative SQL-on-Hadoop systems exist, such as HortonWorks Stinger and Cloudera Impala, Hive is a pioneer system that provides SQL-like analysis of HDFS data. We have used hive for our experimentation presented.

The data storage formats mentioned in Introduction section (Text/CSV [9], JSON [10], Avro [11], SequenceFile [12], RCFile [13], ORC file [14], Parquet [15]) have some advantages and disadvantages. Only Avro, SequenceFile, RCFile, ORC file, and Parquet offer compression storage capacity. Furthermore, the Avro and Parquet data formats offer schema evolution. This is the primary reason Avro and Parquet were selected for the studies.

*Avro* is a row-based cross-language file format in Hadoop, a schema-based serialization technique. Avro was created with a primary goal of Schema Evolution, and the schema is segregated from the data, unlike many other traditional file formats. Data can be written with no prior knowledge of schema, and the resulting serialized data is lesser in size as compared to the source. Avro stores schema in JSON format, making it easy to read and interpret by any programming language. Avro creates a binary structured format that is both compressible and splittable. As a result, it can be efficiently used to input Hadoop MapReduce jobs. Avro can capture transactions such as Updates, Inserts, and Delete logs, so it is widely used for OLTP (on-line transaction processing) and has noteworthy performance in Updating, Deleting, Inserting, and querying all columns of rows or rows of the table [4]. Avro can easily handle changes in schema such as addition or changes in existing fields; thus, new programs can scan old data, and old programs can scan new data. Avro is extremely useful in systems where we see data is written continuously.

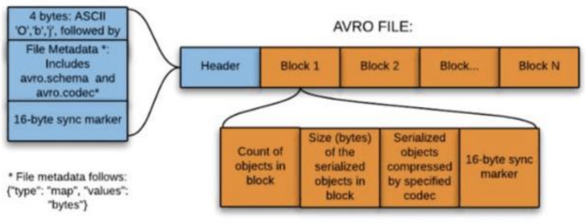
4 bytes: ASCII 'O','b','j', followed by
File Metadata *: Includes avro.schema and avro.codec*
16-byte sync marker
* File metadata follows: {"type": "map", "values": "bytes"}
AVRO FILE:
Header
Block 1
Block 2
Block...
Block N
Count of objects in block
Size (bytes) of the serialized objects in block
Serialized objects compressed by specified codec
16-byte sync marker

**Fig -1**: Avro File Format Structure

*Parquet* is an open-source storage format designed to bring an efficient columnar storage layout in Hadoop. Parquet has a nested structure, and data is populated sparsely. The idea behind columnar format is straightforward: instead of storing the records in row by row, store the record using the column, which is beneficial for analytical processing. Multiple types of data can be converted and stored into Parquet format. Parquet supports well-organized compression and encoding schemes. Parquet supports efficient columnar data representation available to any project in the Hadoop ecosystem, regardless of the choice of data processing framework, data model, or programming language [5]. The main advantage of Parquet is that A query can read and perform on all values for a column while reading only a small fraction of the data from a data file or table.

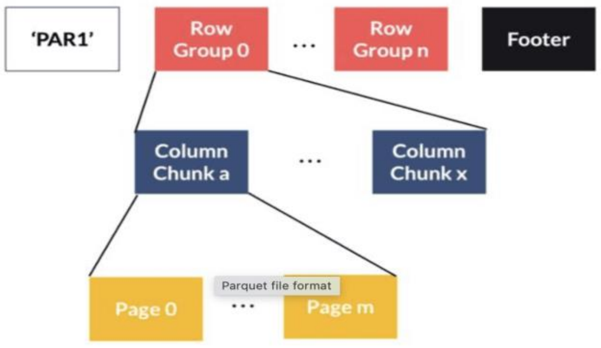

**Fig -2** : Parquet File Format Structure

*ORC, Optimized Row Columnar* is the most widely used file format in Hadoop. It can store data in an optimized way than the other file formats. Original data can be reduced by about 75% due to this; there is an increase in data processing speed and shows better performance than other file formats discussed. An ORC file contains rows of data organized into Stripes and a file footer. When Hive is processing data, the ORC format enhances speed. We are unable to load data into ORCFILE directly. We must first load data into another table then replace it in our freshly formed ORCFILE. It has various advantages over other file formats. It involves storing columns separately, storing statistics (Min, Max, Sum, Count), having a lightweight index, skips blocks of the row that are not part of the query.

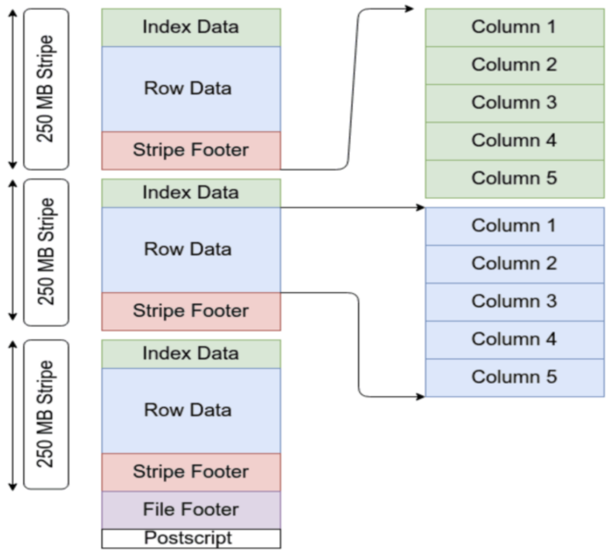

**Fig -3:** ORC File Format Structure

The fundamental question is what are the performance differences between Parquet and Avro in terms of query execution time.

## 3. GOALS AND OBJECTIVES

From the Introduction and Background section of the article, we see how important it becomes to select the right file format to process our jobs, and also see that it will in turn help in processing in terms of speed in the Hadoop ecosystem like Hive. We come up with the task to perform experiments to answer these questions:

**Question 1:** What are the differences in performance (query execution time) between Avro, Parquet and ORC?

**Question 2:** Which data format (Avro, Parquet and ORC) is more compact?

The experiment has been chosen as a research method to address the research questions. There are five steps to the experimenting process. The scope is the first step, followed by planning, execution, analysis, interpretation, and finally reporting. Independent variables have been defined in order to formulate the scope of the experiments. The data format type (Avro / Parquet) has been designated as an independent variable, while performance and compactness have been designated as dependent variables. As a result, the experiment's scope has been defined as follows: Analyze the data formats Avro and Parquet with the purpose of comparing performance and compactness from a researcher's perspective in the context of a Big Data storage format.

The experiments chose Avro and Parquet based on the assumption that Avro supported row-oriented data access should provide better performance on scan queries, e.g. when all columns are of interest, but Parquet format should provide a better performance on column-oriented queries, e.g. when only a subset of those are selected.

After scoping and planning, the operation stage has been performed. Organizing the experiments includes preparation, execution, and data validation tasks that are described in the next section.

## 4. RESEARCH METHODOLOGY

Numerous extensive data management systems are available today, including Oracle's Big Data Appliance, IBM's Apache Hadoop, Cloudera's CDH, Hortonwork's HDP, Apache Spark, etc. These systems are primarily focused on massive data storage and processing, but their techniques may differ. For example, MapReduce's processing method differs from Spark's DAG approach. The HDP Hadoop distribution from Hortonworks is being chosen for this paper. The key reason for this is the platform's widespread popularity due to its openness. More open-source Hadoop ecosystem projects have been merged into Cloudera than any other platform. As a result, it enjoys greater acceptance among businesses because it does not result in vendor lock-in.

### 4.1 Cluster Specifications

These experiments were performed in a Hadoop cluster based on a Horton Works setup. The Horton works version we are using for performing this test is the 2.6 version. This is one of the most stable versions released by Horton. The cluster is being set up in such a way that it can seamlessly support large/text processing. Two nodes name nodes are running in a high-available manner. This is an advisable amount of master nodes recommended by Horton. The remaining 10 data nodes run the worker roles for the Hadoop services. This is an empirically chosen amount of data nodes.

### 4.2 Data used for experiments

Various data formats and data from different databases are being used for this experiment. The data being used was this was scaled for 30 times, this is estimated around to be 300 GB. All the data for this has been sent to the HDFS directory of the Hadoop cluster. As a plan, we have first sent all the data in plain text format and then those data have been converted to different formats. Different tables have different rows of data in them, this experiment will then compare on all the formats presented. Data is being loaded into the hive table using different methods for different file formats. Plain text format creation has been done using the *CREATE TABLE* method with "stored as *TEXTFILE*", the parquet file has been stored using the "stored as *PARQUET*", the files in AVRO format have been stored using "stored as *AVRO*" and finally the files in ORC have been stored using "stored as *ORC*".

### 4.3 Queries Performed

The queries were performed on all the file formats, the various queries written were pickup from git repo [29] with has the necessary combinations for it. The change is connected to the clause "l shipdate = date '1998-12-01' - interval '[DELTA]' day (3)" in the 'where' clause. Because data load into Hive without a workaround approach of at least four steps (create temp table, load data, create a table with correct data types, and insert data therefrom temp table) only supports string type date values, the date interval has been replaced with the exact date and function to date() has been added to return the date from a string type date value stored in Hive table.

## 5. EXPERIMENTAL RESULTS

This section presents the results of the experiments and answers to the research questions.

Loading data into Hadoop and converting it from plain text to Avro and Parquet formats saves a lot of storage space. The same data needs 2 times less storage space in Avro format, and 3 times less in Parquet format, as seen in Fig. 8. This is a response to RQ.2, the second research question: Which data format is more compact (Avro or Parquet)? As a result, Parquet has a smaller footprint than Avro. Despite the fact that Avro and Parquet both use the Snappy compression algorithm, the difference between the two demonstrates that Parquet is roughly 1.5 times more compact than Avro.

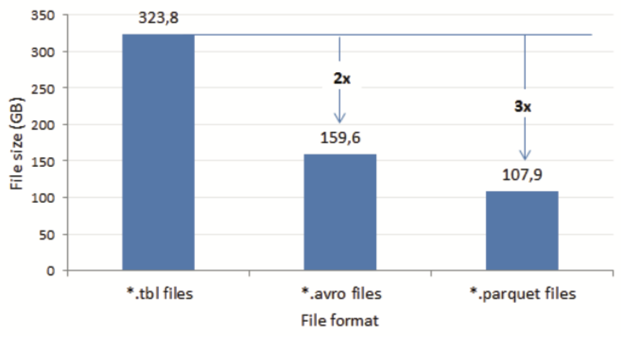

**Chart -1**: File Compression across File Formats

Now we look into the aspect of how data is being stored into tables of different data formats, we have inserted a different number of records into all the tables and have seen that ORC file format performs better in terms of data insertion. The operational time taken is quite less as compared to the rest file formats that we have discussed.

**TABLE -1:** File Loading in Different Across File Formats

| No. of records | Parquet (sec) | Avro (sec) | ORC (sec) |
|---|---|---|---|
| 50000 | 5.51 | 10.09 | 1.7 |
| 100000 | 7.02 | 15.98 | 2.34 |
| 200000 | 12.84 | 26.75 | 5.73 |

Performance Comparison Results of the file sizes across different formats. Notice the vast variation in file sizes for 500 million rows. JSON has the largest footprint whereas Parquet has the lowest.

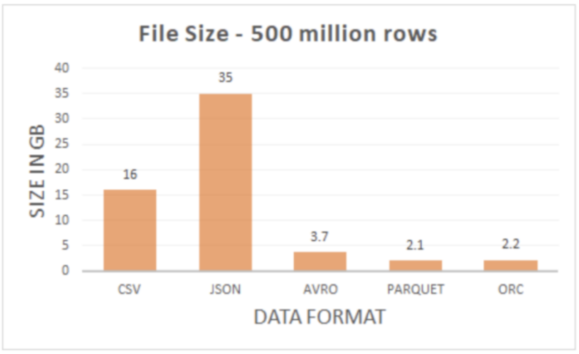

**Chart -2**: Size comparison of File Formats

The outcomes of the grouping process and determining the maximum value across several formats are shown below. It's important to note that this is merely a mathematical process. It's very similar to a Data Analytics use case. I'm pleased to report that all binary formats (AVRO, Parquet, and ORC) performed admirably. Parquet and ORC were nearly identical in terms of performance, although ORC had a smaller file footprint. It comes as no surprise that ORC adoption has plummeted in recent years. I barely ever come across projects in ORC format these days.

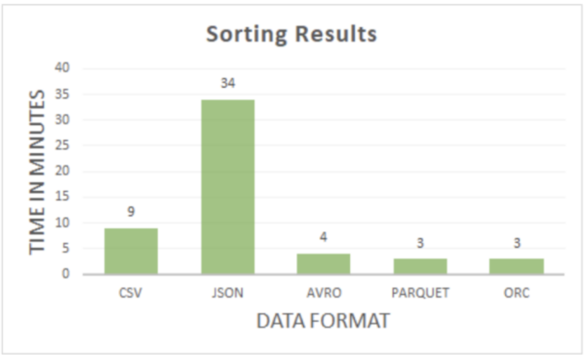

**Chart -3**: Performance comparison on querying

**TABLE -2**: Query execution time (seconds) across file formats

| Query | Data Format | | | |
|---|---|---|---|---|
| | Textfile | Avro | Parquet | ORC |
| Query 0 | 132 | 200 | 34 | 28 |
| Query 1 | 306 | 320 | 142 | 122 |
| Query 2 | Failed | Failed | Failed | Failed |
| Query 3 | 430 | 500 | 277 | 230 |
| Query 4 | 350 | 390 | 200 | 170 |
| Query 5 | 500 | 550 | 320 | 280 |
| Query 6 | 145 | 230 | 64 | 44 |
| Query 7 | 633 | 640 | 436 | 380 |
| Query 8 | Failed | Failed | Failed | Failed |
| Query 9 | Failed | Failed | Failed | Failed |
| Query 10 | 403 | 460 | 230 | 180 |
| Query 11 | 325 | 310 | 270 | 213 |
| Query 12 | 325 | 360 | 180 | 167 |
| Query 13 | 216 | 244 | 200 | 162 |
| Query 14 | 275 | 315 | 150 | 118 |
| Query 15 | 608 | 675 | 325 | 299 |
| Query 16 | 280 | 300 | 238 | 198 |
| Query 17 | 600 | 700 | 344 | 312 |
| Query 18 | 680 | 800 | 428 | 409 |
| Query 19 | Failed | Failed | Failed | Failed |
| Query 20 | 540 | 645 | 390 | 365 |
| Query 21 | 1000 | 1266 | 670 | 634 |
| Query 22 | 210 | 300 | 150 | 134 |
| Query 23 | 28 | 55 | 25 | 18 |

## 6. CONCLUSIONS

To avoid repeat and to owe to space limits, some of the final conclusions are offered in this part, which is far from a full review. The studies in this article were based on a thorough examination of SQL-on-Hadoop using small data formats [6]. In order to answer the research concerns about Parquet and Avro format, a gap and the need for additional experiments and investigations have been identified as a consequence of a comprehensive literature assessment. Because of both design specifics, none of the 17 studies reviewed at the end of the systematic literature review have a direct focus on comparing three binary data storage formats – Parquet, Avro, and ORC.

The experiments reveal that using Avro is only beneficial in terms of saving storage space. Even queries from Textfile format tables are slower than queries from Avro tables. GIT queries from ORC format tables, on the other hand, provide a significant performance improvement over Textfile, Parquet and Avro queries. When compared to others, ORC can deliver a 2x faster execution time on average. There isn't much of a difference between the scan and aggregation queries offered.

Experiments with GIT datasets have gotten a lot of attention. GIT decision support benchmarks are frequently utilised in relational database system performance evaluation nowadays. Because DBGEN allows for datasets with scale factors greater than 1TB, GIT datasets can be used to evaluate the performance of Big Data management systems. In the future, query performance could be measured using the TPC-DS standard benchmark, which is more appropriate for Big Data systems. Other query engines and frameworks such as Impala, HAWQ, IBM Big SQL, Drill, Tajo, Pig, Presto, and Spark, Cascading, Crunch could also be investigated for additional experiments to obtain more detailed experience with tiny data formats.

## REFERENCES

[1] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[2] Shuo Zhang, Li Miao, Dafang Zhang, "A Strategy to Deal with Mass Small Files in HDFS" 2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics.

[3] Daiga Plase, Laila Niedrite, Romans Taranovs, "Accelerating Data Queries on Hadoop Framework by Using Compact Data Formats". 2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)

[4] M. Sharma, N. Hasteer, A. Tuli and A. Bansal, "Investigating the inclinations of research and practices in hadoop: A systematic review," Confluence the Next Generation Information Technology Summit (Confluence), in Proc. of 5th International Conference - IEEE, 2014, pp. 227-231.

[5] Gartner Says Smartphone Sales Surpassed One Billion Units in 2014. [Online]. Available: http://www.gartner.com/newsroom/id/2996817.

[6] D. Plase, "A systematic review of SQL-on-Hadoop by using compact data formats," Preprint. [Online]. Available: https://dspace.lu.lv/dspace/handle/7/34452. [Accessed: 2-Nov-2016].

[7] Y. Chen, X. Qin, H. Bian, J. Chen, Z. Dong, X. Du and H. Zhang, "A study of sql-on-hadoop systems," in Proc. of Workshop on Big Data Benchmarks, Performance Optimization, and Emerging Hardware, Springer International Publishing, 2014, pp. 154-166.

[8] K. Shvachko, H. Kuang, S. Radia and R. Chansler, "The hadoop distributed file system," in Proc. of IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 2010, pp. 1-10.

[9] CSV files. [Online]. Available: https://tools.ietf.org/html/rfc4180.

[10] JSON specification. [Online]. https://tools.ietf.org/html/rfc7159. [Accessed: 1-Nov-2016].

[11] Avro specification. [Online]. http://avro.apache.org/docs/current/spec.html [Accessed: 1-Nov-2016].

[12] Sequence File documentation. [Online]. Available: https://wiki.apache.org/hadoop/SequenceFile [Accessed: 1-Nov-2016].

[13] Y. He, R. Lee, Y. Huai, Z. Shao, N. Jain, X. Zhang and Z. Xu, "RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems," in Proc. of IEEE 27th International Conference on Data Engineering (ICDE), 2011, pp. 1199-1208.

[14] ORC Files. [Online]. Available: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC

[15] Parquet official documentation. [Online] https://parquet.apache.org/documentation/latest

[16] ApacheThrift.[Online].Available: http://thrift.apache.org

[17] N. Palmer, E. Miron, R. Kemp, T. Kielmann and H. Bal, "Towards collaborative editing of structured data on mobile devices," in Proc. of 12th IEEE International Conference on Mobile Data Management (MDM), vol.1, 2011 (June), pp. 194-199.

[18] S. Zhang, L. Miao, D. Zhang and Y. Wang, "A strategy to deal with mass small files in HDFS," in Proc. of IEEE Sixth International Conference In Intelligent Human-Machine Systems and Cybernetics (IHMSC), vol. 1, 2014 (August), pp. 331-334.

[19] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira and P. O'Neil, "C-store: a column-oriented DBMS," in Proc. of the 31st international conference on Very large data bases, VLDB Endowment, 2005, pp. 553-564.