Comparative Characteristics of Big Data Storage Formats

Vladimir Belov¹,², Andrey Tatarintsev¹ and Evgeny Nikulchev¹

¹ MIREA – Russian Technological University, Moscow 119454, Russian Federation
² Russian Academy of Education, Moscow 119121, Russian Federation

e-mail: belov_v.a@mail.ru, tatarintsev@mirea.ru, nikulchev@mail.ru

Abstract. One of the most important tasks of any platform for big data processing is the task of storing data received. Different systems have different requirements for the storage formats of big data, which raises the problem of choosing the optimal data storage format to solve the current problem. This paper describes the five most popular formats for storing big data, presents an experimental evaluation of these formats and a methodology for choosing the format.

1. Introduction

The growth in research in the development of big data platforms and systems has led to the emergence of a variety of software products and tools [1–3], which, in turn, has led to the emergence of a large number of data storage formats. In this regard, when developing platforms and analytical systems, the question arises of choosing the use of a particular format for storing data. In the past few years, Hadoop system has become de facto the standard of platforms for big data processing and storing [4]. Hadoop is a set of software utilities, the core of which is a distributed file system that stores data in certain formats, and a data processor that implements the MapReduce processing model [5]. However, due to various limitations of this system [6], in addition, new implementations of big data processing systems were implemented (eg., Spark [7]), which, on the one hand, are independent products, and on the other hand, are additional tools for the Hadoop system. One of the most important tasks of any platform for big data processing is the task of the storing data received. In recent years, NoSQL technology has become increasingly popular [8], the storage of data in which is different from traditional SQL databases [8], and each type of NoSQL databases imposes its own requirements on the way of data storage.

This paper describes the study carried out on five data storage formats, specifically avro, csv, json, orc and parquet. Each of these formats is popular when using the Hadoop system as the basis for building own data processing and storage platform. The aim of the paper is to study the features of file formats used for storing big data, as well as to conduct an experimental evaluation of the formats.

The article is organized as follows. The second section provides an overview of data storage formats, focusing on the formats tested in this study. In the third section, the details of the experimental setup are described, the configuration of the hardware, software, and the data preparation method is given, and the
results of the experiment are described. The fourth section presents an algorithm for finding the optimal data storage format for a specific task. At the end, the results of the study are presented.

2. Storage formats review
Let us consider the features of the internal structure of the studied data storage formats.

Avro is a row-oriented data storage format. The main feature of the format is the presence of a schema in the JSON format, which allows faster reading and interpretation operations [9]. The file structure consists of a header and data blocks [9]. Avro is not a strongly typed format: the type information of each field is stored in the metadata section along with the schema. Avro format supports primitive (null, boolean, int, long, float, etc.) and complex composite (array, map) types.

Comma-Separated Values is a textual format that describes data in tabular form. The structure of the CSV file is represented as strings separated by commas. A header is assumed, but this is not a strict requirement. CSV file does not support different data types and structures - all data is presented as strings.

JavaScript Object Notation is a simple text format based on a subset of the JavaScript programming language. With the growing popularity of NoSQL solutions, the JSON has gained popularity in storing big data in document databases [8]. The structure of a JSON file consists of two parts: a collection of key/value pairs and an ordered list of values. JSON supports data types and structures such as string, number, boolean, arrays, null, internal objects.

Optimized Row Columnar is a column-oriented storage format in the Apache Hadoop ecosystem [7]. Data in ORC is strongly typed, therefore, when writing, an encoding is chosen that is most suitable for each data type [B2]. ORC has a shared internal structure - division into strips independent from each other. The metadata, which constitutes an essential part of the file, is stored in a compressed form, and includes [B3] statistical and descriptive information, indexes, stripe and stream information. Indexes are built on each of the columns, which affects the reading speed by increasing the size [10]. Column values are stored in a compressed form, making it possible to read and decompress only the data block that is needed. ORC supports a complete set of types, including complex types (structures, lists, maps, and unions) [10]. ORC also complies with ACID requirements by adding delta files.

Apache Parquet is a column-oriented binary format that takes advantage of the compressed information presentation. Parquet allows to define compression schemes at the column level and add new encodings as they appear [C1]. Parquet files have a layered structure [11]. Parquet supports simple (boolean, int32, float etc.) and complex (byte_array, map) data types [11]. The Parquet format explicitly separates metadata from data, allowing columns to be split across multiple files, as well as having a single metadata file referencing multiple parquet files. Metadata is written after meaningful data to provide a one-pass write.

3. Experimental part
To estimate the data storage formats, a study was carried out consisting of two parts:
1. Comparative analysis of the main characteristics of data storage formats;
2. Experimental evaluation of the studied data storage formats.

3.1. Comparative analysis
For a comparative analysis, the following characteristics were selected, describing the features of each data storage format: platform independence, the ability to change the file, the ability to record complex structures (eg., lists, dates, internal objects), compliance with ACID requirements [12], format type (internal file structure), file compression, the presence of metadata in the file of the studied format.

The results of the analysis are presented in Table 1.
### Table 1. Comparative analysis of the main characteristics of data storage formats

|                        | avro | csv | json | orc | parquet |
|------------------------|------|-----|------|-----|---------|
| Platform independence  | +    | +   | +    | –   | –       |
| The ability to change the file | –    | +   | +    | –   | –       |
| Record complex structures | +    | –   | +    | +   | +       |
| Compliance with ACID   | –    | –   | –    | +   | –       |
| Format type            | row-oriented | text, string | text, objective | column-oriented | column-oriented |
| File compression       | +    | –   | –    | +   | +       |
| The presence of metadata | –    | –   | –    | +   | +       |

#### 3.2. Experimental evaluation

The second stage in the study was to conduct an experimental evaluation of these formats. The experimental evaluation consisted of simulated processing of the dataset. An experimental stand was deployed for testing. Table 2 shows the configuration of the formed stand.

For the study, a dataset of 10 million records was generated. The description of the generated data is presented in Table 3.

### Table 2. Experimental stand configuration

| Element               | Characteristics                      |
|-----------------------|--------------------------------------|
| CPU                   | Intel Core i7-8565U 1.8 GHz 4 cores  |
| RAM                   | 16 GB                                |
| Operating system      | Windows 10 64x                       |
| Platform              | Java Virtual Machine                |
| Programming language used | Java v. 1.8                         |
| The framework used    | Apache Spark v. 2.4                  |

### Table 3. Description of the generated data

| Field name   | Data type     |
|--------------|---------------|
| name         | string        |
| surname      | string        |
| age          | 32 bit integer|
| country      | string        |
| balance      | 64 bit integer|
| card number  | string        |
| currency     | string        |
| account open date | calendar     |

The figure 1 illustrates an experiment schema. The host file system contains the generated dataset. A Java Virtual Machine which supports the Spark application executor (driver) is installed on the host. When the driver starts, a Spark Context is formed, which is necessary for the further operation of the application. After loading, the application reads the generated data from the file system one by one and measures the time from the start of reading to the end of the next operation. Since a feature of a Spark application is the presence of lazy evaluations [13] in it, that means that any transformations are not performed until these transformations are acted upon, then the moment of completion of the operation is the reading of the first data set (the `take(1)` operation), unless otherwise provided by the research method. Upon completion of the operation on all datasets, the application saves the results in a text file.
For each data format, a study was conducted, consisting of test runs of the Spark application and performing the same set of operations. Next, consider each test and the results obtained.

**Total dataset volume.** One of the most important characteristics of data is its volume. Since volume becomes important in systems for processing and storing big data, it becomes necessary to search for such a format that would have the ability to store data with a minimum volume. Figure 2 illustrates the total size of the generated dataset for each format. The histogram shows that the json data format showed the worst result. This indicator is explained by the fact that json, firstly, is a string representation of an object, and secondly, it contains not only the values of the object's fields, but the names of the fields themselves in the body of each object.

**Reading all lines.** The most important parameter in data processing and analysis is the time to read the data. In this test, the time taken to read all objects was measured. The results shown in figure 3 showed that most formats performed approximately the same.

**Data filtering.** Data filtering is one of the most frequently used operations in data processing and analysis. In this test, two parameters were filtered. The results of the launches are shown in figure 4. In this case, json again showed the worst results. It should be noted that the avro and csv formats worked twice as fast, and orc and parquet more than 10 times faster, which is explained by the presence of metadata storing some of the information about the data contained in the files.

**Search for unique objects.** An equally important operation in data processing and analysis is the search for unique objects, known in the SQL language as the DISTINCT operation. In this test, all formats showed approximately equal results. Figure 5 illustrates the test results.

**Sorting.** Sorting is the most complex operation, both in development and in databases, so the results of this test are important when analyzing big data storage formats. Figure 6 shows a histogram of the execution time of sorting by a string field. The results show that the worst time was shown by the json format, the other formats showed approximately the same time.

**Grouping.** Grouping is also one of the most commonly used operations in data analysis and processing. The results of measuring the execution time for each format are shown in figure 7. The histogram shows that the orc and parquet formats showed the best results, which is explained by the presence of metadata on the data contained in the files. The json format showed the worst result. This is because the application has to parse the next object contained in the file each time.

The worst result in all tests was shown by the json format. According to the results of the study, it follows that this type is well suited for storing single objects or data that require periodic changes. Also,
this type is suitable in platform independent systems. The format is easy to read, thus, it can be used by specialists not associated or indirectly associated with software development.

The avro and csv formats showed average results. However, the avro format is not as popular in the big data industry as the csv format. With similar test results, the csv format, in contrast to the avro format, is more convenient to use since it allows changing and reading the contained data without using specialized software.

![Figure 2. Total dataset volume, MB](image1)

![Figure 3. Reading all lines, sec](image2)

![Figure 4. Data filtering, ms](image3)

![Figure 5. Search for unique objects, ms](image4)

![Figure 6. Sorting, ms](image5)

![Figure 7. Grouping, ms](image6)

The best results were shown by two formats: orc and parquet. Both formats are platform dependent. To work with them, it is required to have a cluster with the Hadoop platform deployed on it or to have deployed Spark executor. Each of these types is a compressed data format that contains metadata that stores statistics and information for quick access to data. However, both formats are inferior to formats like json and csv in terms of changeability. Both formats support the WORM (“write once - read many”) principle [14] of data storing. However, the orc format, in contrast to the parquet format, supports the ACID technology by adding files containing information about updating or changing data, which affects its performance in a few tests.

4. Choosing a solution

The problem of choosing the optimal format was presented in the form of several optimization tasks using tropical optimization algorithm [15]. The algorithm for finding the optimal solution consists of two stages: paired comparisons of each format according to the criteria and the comparison of the criteria themselves. The second stage, used for finding optimal format, is described in detail in [15].

Let $A_{ij}$ are matrices of paired comparisons of alternatives by $n$ criteria, and $C$ - matrix of paired comparisons of the criteria themselves.

To compare the storage formats, it is required to introduce several comparison rules:

1. Since the results obtained experimentally are inversely proportional to the criteria for the success of the test, the degree of preference for alternative $i$ over alternative $j$ is calculated by the formula

$$a_{ij} = \frac{a_j}{a_i}$$  \hspace{1cm} (1)
The assessment of the preference of the criteria and the assessment based on the indicators obtained on the comparative characteristic are calculated according to the formula given in [12, 15]:

\[ a_{ij} = \frac{a_i}{a_j} \]  

(2)

2. To determine preferences in the comparison matrices according to the main characteristics of the formats and the comparison matrix of the criteria themselves, it is necessary to introduce the requirements of the data storage subsystem being developed, according to which one or another format should be preferred.

3. Since the results contain a large range that can affect the calculation of the optimal format, it was decided to bring the results to certain ranges according to the following algorithm:
   - take the minimum and maximum value from the array of results for a specific criterion;
   - divide the resulting range into \( n \) segments, where \( n \) is a number of alternatives;
   - when the result falls into the range with the minimum values, we assign it \( n \) points, respectively, for the format with the maximum value -1 point;
   - when setting preferences, we act according to formula (2), where \( a_i \) and \( a_j \) – are the assigned points for the \( i^{th} \) and \( j^{th} \) format, respectively.

To determine the matrices according to the criteria of the main characteristics and the matrix of the preference of the criteria, it is needed to introduce the requirements that the data storage format must meet. It is important to note that these requirements are individual for each project. Thus, all of the following description is an example of calculating the optimal format. We introduce the following rules for choosing preferences:

1. Platform independence is not the most important characteristic, since most systems for working with big data are developed on the basis of ready-made platform solutions [16].
2. The ability to record complex structures plays an important role, since it provides great opportunities for data processing and analysis.
3. The ability to modify data and comply with ACID requirements is not critical, since most big data storage platforms comply with the WORM principle.
4. The type of format and the possibility of compression when working with big data plays an indirect role, since it affects, first of all, the volume of data that is obtained experimentally.
5. The presence of metadata, as in the previous paragraph, is an indicator that does not require analysis, since it affects the speed of reading and grouping data, which is obtained experimentally.

According to the indicators obtained experimentally, we will be guided by the following rules:

1. The data volume plays an important role in the processing and storage of big data, but is not critical, since the storage hardware has become much cheaper in recent years.
2. Reading all lines is an important indicator, since it most fully reflects the speed of data processing using a particular data storage format.
3. The filter and search for unique values are equally important characteristics, however, these functions rely on the subtraction of all strings, the importance of which is defined in the previous paragraph.
4. Applying a function, grouping and finding the minimum value are the next most important indicators, since they are interesting from the point of view of analytics than engineering.
5. Sorting is the least important of the criteria presented, as it is most often used to visualize data.

To assess the preference of one or another indicator, the following scale is introduced:

- equals = 1;
- more (less) important = 2 (1/2);
• much more important = 4 (1/4);
• critical = 6 (1/6);
• in some cases, it is possible to use intermediate values.

Taking into account all the rules described above, we obtain the matrices of pairwise comparisons for each criterion.

The criteria preference matrix looks like the following, see Table 4

| Criteria                     | Platform independence | Recording complex structure | Volume | Reading all lines | Filter | Unique values | Applying function | Grouping | Min value | Sorting |
|------------------------------|----------------------|-----------------------------|--------|------------------|--------|---------------|-------------------|----------|-----------|---------|
| Platform independence        | 1                    | 1/2                         | 1/2    | 1/5              | 1/5    | 1/4           | 1/4               | 1/4      | 1/4       | 1/2     |
| Recording complex structure  | 2                    | 1                           | 1/4    | 1/4              | 1/2    | 1/2           | 1/2               | 1/2      | 1/2       | 1       |
| Volume                       | 2                    | 1                           | 1/4    | 1/4              | 1/2    | 1/2           | 1/2               | 1/2      | 1/2       | 2       |
| Reading all lines            | 5                    | 4                           | 4      | 1                | 1/2    | 1/2           | 1/2               | 1/2      | 1/2       | 2       |
| Filter                       | 5                    | 4                           | 4      | 1                | 1/2    | 1/2           | 1/2               | 1/2      | 1/2       | 2       |
| Unique values                | 4                    | 2                           | 2      | 2                | 1      | 1             | 1                 | 1        | 1         | 4       |
| Applying function            | 4                    | 2                           | 2      | 2                | 2      | 1             | 1                 | 1        | 1         | 4       |
| Grouping                     | 4                    | 2                           | 2      | 2                | 2      | 1             | 1                 | 1        | 1         | 4       |
| Min value                    | 4                    | 2                           | 2      | 2                | 2      | 1             | 1                 | 1        | 1         | 4       |
| Sorting                      | 2                    | 1                           | 1      | 1/2              | 1/2    | 1/4           | 1/4               | 1/4      | 1/4       | 1       |

Using the given matrices, calculate the rating vector of alternatives [15].

1. According to criteria matrix, calculate the weight vector of criteria:

\[ w = (\mu^{-1}C)^* \text{, where } \mu = tr \ C \oplus ... \oplus tr \ C^m \text{ and } C \text{ is criteria matrix} \]

2. If result matrix contains more than one vector (up to a positive factor), find the least differentiating vectors:

\[ w_1 = (\delta^{-1}11^T \oplus \mu^{-1}C)^* \text{, where } \delta = 1^T (\mu^{-1}C)^* 1 \]

and the most differentiating vectors:

\[ w_2 = P (I \oplus P_{sk}P) \]

where \( P \) is a matrix \((\mu^{-1}C)^*\) removing columns linearly independent from another, \( P_{sk} \) is a matrix created from matrix \( P \) by nullifying every element except \( p_{sk}, \) and \( k \) and \( s \) indexes are calculated using following formula:

\[ k = \arg \max_j 1^T p_j p_j^* 1, \quad s = \arg \max_i p_{ik}^{-1}. \]

3. Using \( w_1 = (w_1^{(1)}) \) and \( w_2 = (w_1^{(2)}) \) calculate weighted amounts of paired comparisons matrixes:

\[ D_1 = w_1^{(1)} A_1 \oplus ... \oplus w_m^{(1)} A_m, \quad D_2 = w_1^{(2)} A_1 \oplus ... \oplus w_m^{(2)} A_m \]
4. Calculate the least differentiating vector of the rating of alternatives:

\[ x_1 = (v_1^{-1}D_1)^*, \text{ where } v_1 = \operatorname{tr} D_1 \oplus \ldots \oplus \operatorname{tr}^\mu(D_1^n) \]

If resulting vector is not unique, calculate it in the different way:

\[ x_1 = (\delta_1^{-1}1^T \oplus v_1^{-1}D_1)^*, \text{ where } \delta_1 = 1^T(v_1^{-1}D_1)^*1 \]

5. Calculate the most differentiating vector of the rating of alternatives:

\[ x_2 = (v_2^{-1}D_2)^*, \text{ where } v_2 = \operatorname{tr} D_2 \oplus \ldots \oplus \operatorname{tr}^\mu(D_2^n) \]

If resulting vector is not unique, calculate it in the different way:

\[ x_2 = Q((I \oplus Q_{sk}Q)^*)^*, \]

where Q is a matrix \((v_2^{-1}D_2)^*\) removing columns linearly independent from another, \(Q_{sk}\) is a matrix created from matrix Q by nullifying every element except \(q_{sk}\), and k and s indexes are calculated using following formula:

\[ k = \arg \max_i 1^T q_i q_i^{-1}, \quad s = \arg \max_i q_{sk}^{-1}. \]

To facilitate calculations, the Jupyter Notebook script using programming language Python was written. The results of calculating are presented in followings.

At first, calculate spectral radius using calculation rules in the independent semifield - \(\mu = 1.5874\) and w looks like the following numpy array:

\[
\begin{bmatrix}
1 & 0.5 & 0.5 & 0.19843 & 0.19843 & 0.15749 & 0.15749 & 0.15749 & 0.15749 & 0.39685 \\
1.25993 & 1.00001 & 1.00001 & 0.39685 & 0.39685 & 0.31498 & 0.31498 & 0.31498 & 0.31498 & 0.7937 \\
1.25993 & 1.00001 & 1.00001 & 0.39685 & 0.39685 & 0.31498 & 0.31498 & 0.31498 & 0.31498 & 0.7937 \\
3.17482 & 2.51985 & 2.51985 & 1.00001 & 1.00001 & 0.7937 & 0.7937 & 0.7937 & 0.7937 & 2.00001 \\
3.17482 & 2.51985 & 2.51985 & 1.00001 & 1.00001 & 0.7937 & 0.7937 & 0.7937 & 0.7937 & 2.00001 \\
4.00002 & 3.17482 & 3.17482 & 1.25993 & 1.25993 & 1.00001 & 1.00001 & 1.00001 & 1.00001 & 2.51985 \\
4.00002 & 3.17482 & 3.17424 & 1.25993 & 1.25993 & 1.00001 & 1.00001 & 1.00001 & 1.00001 & 2.51985 \\
4.00002 & 3.17482 & 3.17482 & 1.25993 & 1.25993 & 1.00001 & 1.00001 & 1.00001 & 1.00001 & 2.51985 \\
1.25992 & 0.7937 & 0.7937 & 0.31498 & 0.31498 & 0.25 & 0.25 & 0.25 & 0.25 & 1 \\
\end{bmatrix}
\]

To find the least differentiating weights vector, let us calculate weights vector. The result gives the following the least differentiating weights vector:

\[
w_1 \approx \left( 1 \begin{bmatrix} 1.26 & 1.26 & 3.17 & 3.17 & 4 & 4 & 4 & 1.26 \end{bmatrix}^T \begin{bmatrix} 0.63 & 0.8 & 0.8 & 2 & 2 & 2.5 & 2.5 & 2.5 \end{bmatrix} \right)
\]

The resulting matrix contains to vectors. For following calculation, we choose only one vector – for example, the first one. Using the weights vector, let us calculate the least differentiating vector of rating of alternatives:

\[ x_1 \approx (1 1 0.56 1.42 1.42)^T \]

Let us calculate the most differentiating vector of rating of alternatives. At first, calculate weights vector for this:
For the example, we take only first weights vector. Let us calculate the most differentiating vector of rating of alternatives. As a result, the following vector was obtained:

\[ w_2 \approx \begin{pmatrix} 1 \\ 0.4 \\ 2 \\ 2 \\ 5 \\ 0.8 \\ 5 \\ 2.5 \\ 6.35 \\ 2.5 \\ 6.35 \\ 2.5 \\ 6.35 \end{pmatrix}^T \]

The resulting vector looks similar to the previous one. According to this decision, the format rating is built as follows:

parquet ≅ orc > avro ≅ csv > json.

The parquet and orc formats received the highest score in the ranking of alternatives. The avro and csv formats showed an average result. Json has the worst result.

5. Conclusion
The paper described the study of five most popular formats for storing big data. The study consisted of two parts: theoretical and experimental. At the first stage, a comparative analysis of the main characteristics of the studied formats was carried out; at the second stage, an experimental evaluation of these formats was prepared and carried out. For the experiment, an experimental stand was deployed with tools for processing big data installed on it. The aim of the experiment was to find out such characteristics of data storage formats as the volume and processing speed for different operations using the Apache Spark framework.

In addition, within the study, an algorithm for choosing the optimal format from the presented alternatives was developed using tropical optimization methods, the essence of which is to solve a multicriteria decision-making problem, which results is presented in for of vector of preference degrees. The problem of analysis of the presented solutions is considered. It should be noted that the presented algorithm helps to find the optimal solution for the specific requirements of the system. The described analysis is an example of using the algorithm to solve similar problems.

It is important to note that this result is not a standard, but only an example of calculating the optimal value based on built preferences.

Acknowledgments
This research is funded by the Ministry of Science and Higher Education of the Russian Federation.

References
[1] Gusev A, Ilin D, Kolyasnikov P and Nikulchev E 2020 Engineering Letters 28(2) 420–427
[2] Gusev A, Ilin D and Nikulchev E 2020 Data 5(3) 59
[3] Petushkov G V 2020 Russian Technological Journal 8(1) 21–26 (doi: 10.32362/2500-316X-2020-8-1-21-26)
[4] Shahverdi E, Awad A and Sakr S. 2019 Big Stream Processing Systems: An Experimental Evaluation. DASC 2019 workshop, 35th IEEE International Conference on Data Engineering (ICDEW) 53–60 (doi: 10.1109/ICDEW.2019.00-35)
[5] Apache 2008 MapReduce Tutorial [online] Available: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
[6] Sakr S, Liu A and Fayoumi A G 2013 ACM Computing Surveys 46(1) 11
[7] Zaharia M, Chowdhury M, Franklin M J, Shenker S and Stoica I 2010 Spark: Cluster Computing with Working Sets HotCloud'10: Proceedings of the 2nd USENIX conference on Hot topics in cloud computing 95

[8] Moniruzzaman A B M and Hossain S A 2013 International Journal of Database Theory and Application 6(4) 1–13

[9] Apache. 2012 Avro specification Available: http://avro.apache.org/docs/current/spec.html

[10] ORC. 2020. ORC specification Available: https://orc.apache.org/specification/ORCv1/

[11] Apache 2018 Parquet official documentation Available: https://parquet.apache.org/documentation/latest/

[12] Gray J N 1981 The Transaction Concept: Virtues and Limitations Proc. 7th International Conference on Very Large Data Bases

[13] Naik P G and Naik G R 2020 Big Data Tools: Which, When and How? vol III (Shashwat Publication)

[14] Apache 2018 HDFS Architecture Available: https://hadoop.apache.org/docs/r3.0.1/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

[15] Krivulin N 2016 Methods of tropical optimization in rating alternatives based on pairwise comparisons. Operations Research Proceedings 85–91 (doi: 10.1007/978-3-319-55702-1_13)

[16] Wróbel G and Wikira M D 2019 Journal of Computer Sciences Institute 13 283–287 (doi: 10.35784/jcsi.1296)