symmetry

Article

Choosing a Data Storage Format in the Apache Hadoop System Based on Experimental Evaluation Using Apache Spark

Vladimir Belov [1,2], Andrey Tatarintsev [3] and Evgeny Nikulchev [1,*]

[1] Department of Intelligent Information Security Systems, MIREA—Russian Technological University, 119454 Moscow, Russia; belov_v.a@mail.ru
[2] Data Center, Russian Academy of Education, 119121 Moscow, Russia
[3] Departments of Higher Mathematics 2, MIREA—Russian Technological University, 119454 Moscow, Russia; tatarintsev@mirea.ru
* Correspondence: nikulchev@mail.ru

Abstract: One of the most important tasks of any platform for big data processing is storing the 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. The following data storage formats will be considered: avro, CSV, JSON, ORC, parquet. 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 characteristics of data storage formats, such 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 result of the study is presented in the form of a technique for obtaining a vector of ratings of data storage formats for the Apache Hadoop system, based on an experimental assessment using Apache Spark.

Keywords: formats for storing big data; volume and processing speed data; tropical optimization methods

Citation: Belov, V.; Tatarintsev, A.; Nikulchev, E. Choosing a Data Storage Format in the Apache Hadoop System Based on Experimental Evaluation Using Apache Spark. *Symmetry* **2021**, *13*, 195. https://doi.org/10.3390/sym13020195

Academic Editor: Basil Papadopoulos
Received: 11 January 2021
Accepted: 22 January 2021
Published: 26 January 2021

1. Introduction

The development of technologies that work with data have contributed to the emergence of various tools for big data processing [1]. Big data means such volumes of information collected from various sources, where processing using traditional methods becomes very difficult or impossible [2,3]. At the same time, most researchers agree that big data can be understood through not only the volume, but also their ability to be sources for generating valuable information and ideas [4].

The development of platforms for analytical data processing has become a popular direction in the field of working with big data [5]. Such platforms are designed not only for processing, but also for storing data. The best known among such platforms is Apache Hadoop [6]. Hadoop is a set of software utilities [7], 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 [8].

However, due to various limitations of this system, new implementations of big data processing systems were implemented (e.g., Hive [9], Impala [10], Apache Spark [11], etc.). These tools, on the one hand, are independent products, and on the other hand, are additional tools for the Apache Hadoop system.

Frameworks, such as Apache Spark [12], allow working with a variety of file formats. For this study, five formats supported by this framework were selected: avro, CSV, JSON, ORC, and parquet. 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. Such tools allow providing a convenient language for data selection. In addition, these tools can work with a variety of file formats.

However, when developing system architectures based on Apache Hadoop, the question of choosing the optimal data storage format may arise.

This paper describes the five most popular formats for storing big data in the Apache Hadoop system, presents an experimental evaluation of these formats and tropical optimization methods for choosing an effective solution.

Tropical (or idempotent) mathematics is an area of applied mathematics that studies the theory and applications of semirings with idempotent addition.

The use of models and methods of tropical algebra allows reducing several nonlinear problems to a linear form in terms of an idempotent semiring or semifield. In other words, the transformation of the original problem to the form of tropical optimization with idempotent operations leaves the optimality properties of the original and the reduced problem to be invariant. This means that it is a symmetric transformation. The use of this approach simplifies the interpretation of the results and finds application in solving practical problems of planning, placement, and decision-making.

One of the directions of tropical mathematics is the development of methods for solving optimization problems that can be formulated and solved in terms of idempotent mathematics (tropical optimization problems). The theory is based on the correspondence between constructions over the field of real numbers and similar constructions related to various idempotent semirings. For example, [13] describes the solving problems that reduce to the best approximate solution, in the sense of the Chebyshev metric of a vector linear equation, where the product is understood in the sense of tropical algebra.

The study is aimed at developing a technique that is able to define the most effective data format in the condition described for data format usage in the use of big data.

The article is organized as follows. Section 2 provides background regarding the problem of choosing software components. Section 3 describes the details of the experimental setup. The configuration of the hardware, software, and the data preparation method is given, and the results of the experiment are described. Section 4 presents mathematical methods for choosing the solution based on the tropical optimization theory. Section 5 presents the discussion. The Conclusion presents the results obtained during the study.

2. Background

The problem of choosing software components has been studied by various authors [14–24]. The papers present a selection of various components of the system, as well as methods for the experimental evaluation of the selected components. For example, [14,15] present a methodology for choosing libraries for software development using methods of evolutionary calculus. In [16–18], an experimental assessment of integration messaging systems is presented. This paper presents methodology for conducting a study of the data transfer rate in such systems. However, the authors do not give recommendations on the choice of an integration messaging system.

Papers [19–24] present studies of big data storage formats, such as avro, parquet, orc, etc. These studies represent the results of studying different formats in terms of performance, or choosing an alternative for specific purposes. For example, the authors in [23] study data storage formats for storing data in web systems or data for research in bioinformatics, respectively. These are highly specialized studies for specific tasks. Study [24] addresses a problem similar to the current study. However, this study only affects the avro and parquet storage formats and indirectly talks about other data storage formats.

However, the cited works do not investigate the issue of choosing a data storage format. Most of these studies provide the results of examining each format and recommendations on the choice for the problem under study. It should be noted that data storage can be carried out using relational databases. However, in recent years, NoSQL solutions have gained popularity [25,26], some of which support different data storage formats.

Using a sub-optimal format for storing big data can lead to various errors and difficulties when working with data. Thus, the use of formats that do not support complex data structures (such as arrays or dates) can lead to incorrect result sets when fetching data using SQL-like tools in a system, such as Hadoop. In addition, the use of formats that do not use data archiving or metadata can lead to an increase in data retrieval time. Therefore, for systems where the speed of analytical data processing is critical, a forced delay may occur. For systems that require platform independence, there may be a need for expensive system modifications. Different storage formats for big data affect a number of criteria for software satisfaction. These criteria include the speed of working with data (reading, writing, analytical processing, etc.), the speed of development and implementation, portability to different platforms, etc.

The current study is focused on developing techniques for selecting the optimal storage format for Apache Hadoop. The basis of the proposed technique is an experimental assessment of data storage formats and a mathematical model for choosing the optimal solution based on tropical optimization methods. To select the data format, the paper solves the problem of constructing an assessment system, a system of criteria and methods for obtaining their reliable numerical values based on experimental studies, as well as choosing and using an optimization method based on quality criteria. It should be noted that the study does not solve the issue of the functionality of the proposed formats, but reflects the feasibility of using them in the proposed conditions.

3. Method and Experiment

For the current study, the following data storage formats will be considered: avro, csv, json, orc, parquet.

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

Avro is a row-oriented data storage format. It contains a schema in the JSON format, which allows faster reading and interpretation operations [27]. The file structure consists of a header and data blocks [27]. Avro format supports primitive types, such as Boolean, int, long, float, etc., and complex types, such as array or map.

Comma-separated values (CSV) is a textual format describing data in form of a table. A CSV file does not support different data types and structures—all data are presented as strings.

JavaScript object notation is a simple text format. JSON has gained popularity in storing big data in document databases. JSON supports data types and structures, such as string, number, Boolean, arrays, null, internal objects.

Optimized row columnar is a column-oriented storage format [28]. Data in ORC are strongly typed. ORC has a shared internal structure—division into strips independent from each other. ORC files contain metadata storing in compressed forms, and include statistical and descriptive information, indexes, stripe, and stream information. ORC supports a complete set of types, including complex types (structures, lists, maps, and unions) [29]. ORC also complies with ACID requirements by adding delta files.

Apache Parquet is a column-oriented binary format. It allows defining compression schemes at the column level and adding new encodings as they appear [30]. Parquet supports simple (Boolean, int32, float, etc.) and complex (byte_array, map) data types. The Parquet file contains metadata written after meaningful data to provide a one-pass write.

Table 1 contains the comparation of the described storage formats.

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	-	-	-	+	+

To estimate the data storage formats, the technique described in the following was developed. The technique consists of two parts:

1. Experimental evaluation of the studied data storage formats.
2. Analysis of Spark data processing functions using different storage formats.

3.1. Experimental Evaluation

The first 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. For the study, a dataset of 10 million records was generated. Appendix A contains the experimental resources.

Figure 1 illustrates an experiment schema.

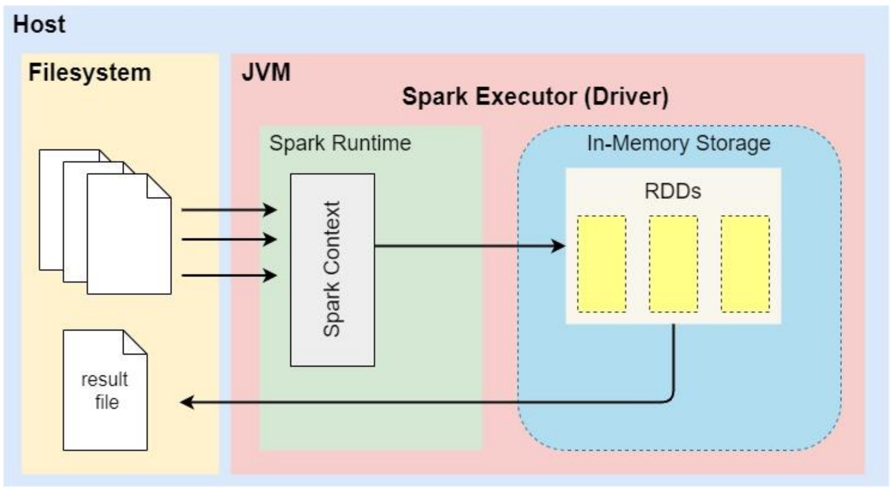

Figure 1. Experiment schema.

The host file system contains the generated dataset. A Java virtual machine, which supports the Spark application executor (driver), is installed on the host. After starting the Spark application, Spark context is generated, the storage files are being read by the Spark application, and the operation studied is being performed. Since the Spark application supports lazy evaluations [31], the moment of completion of the operation is considered to receive the count of the records in the resulting dataset.

For each data format, a study was conducted, consisting of test runs of the Spark application and performing the same set of operations. The following calculations were conducted.

The total size of the dataset. One of the most important characteristics of data is its volume. Since volume becomes critical 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.

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 records was measured.

Filtering data. Data filtering is one of the most frequently used operations in data processing and analysis.

Search for unique strings. An equally important operation in data processing and analysis is the search for unique records.

Sorting. Sorting is the most complex operation, both in design and in databases, so the results of this test are important when analyzing big data storage formats.

Grouping. Grouping is also one of the most used operations in data analysis and processing.

Figures 2–7 illustrate the results obtained during experimental evaluation.

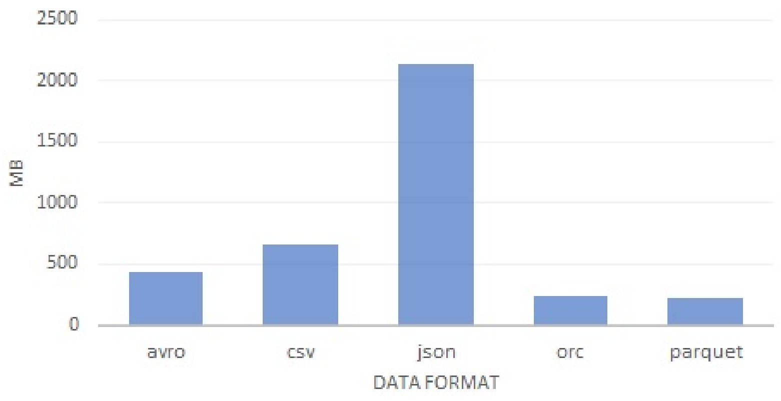

Figure 2. Total dataset volume.

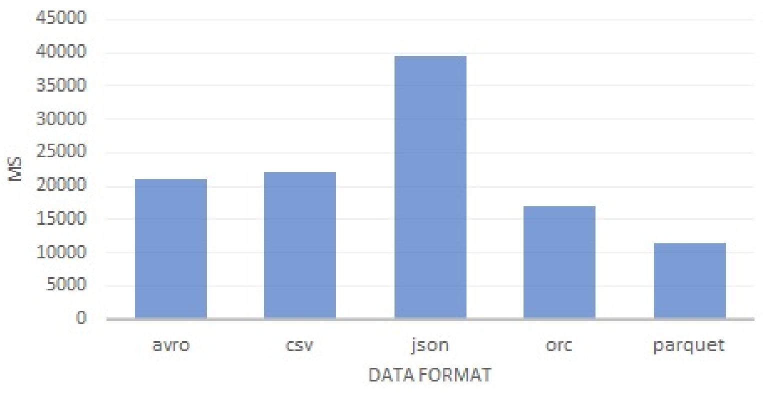

Figure 3. Reading all lines.

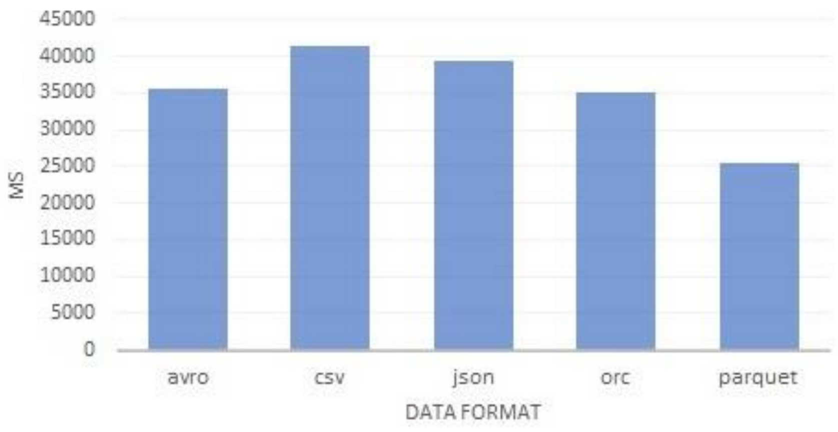

Figure 4. Search for unique objects.

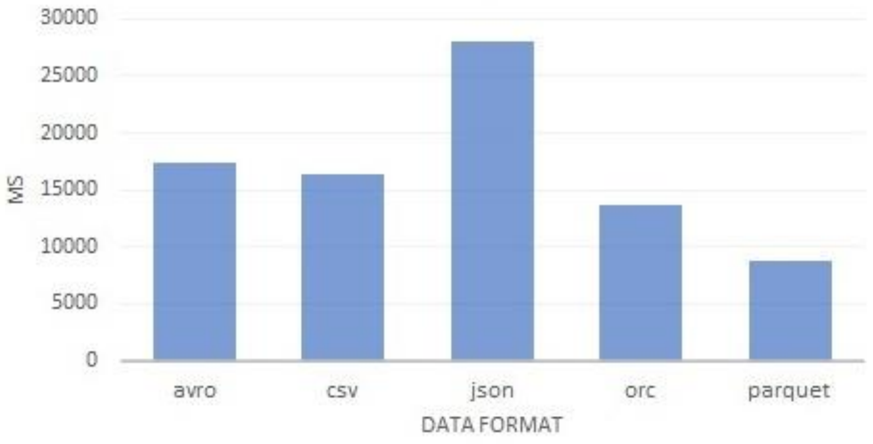

Figure 5. Sorting.

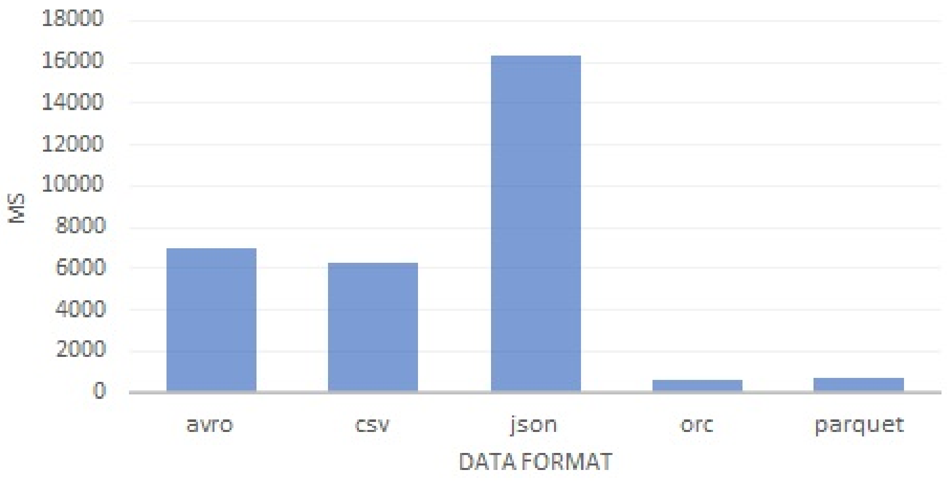

Figure 6. Data filtering.

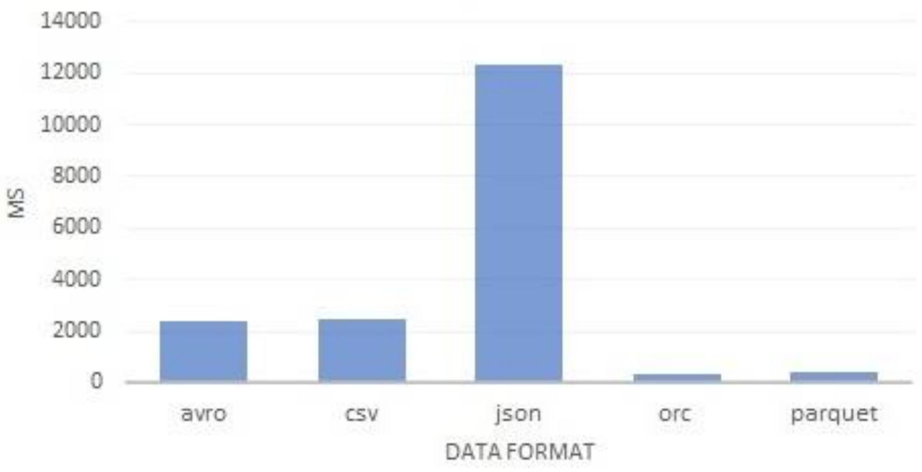

Figure 7. Grouping.

However, the results obtained cannot be considered final, since any data processing and storage system is constantly updated with this data. To study the rate of change in processing time when working with different formats, three more data sets were generated with a similar structure: 5 million records, 25 million records, and 50 million records.

For each of the obtained data sets, the operations described earlier were also carried out. Each of the obtained values was used to calculate the rate of change in the file processing time. The rate was calculated using the following formula:

$$rate_i = \frac{duration_i}{duration_1}$$

where $duration_i$ is an operation duration for i^{th} dataset

Below are graphs of the results of calculating the rate of changes in the processing time of files of different formats, according to operations. The Y-axis on the Figures 8–12 shows the rate calculated for datasets of different volumes.

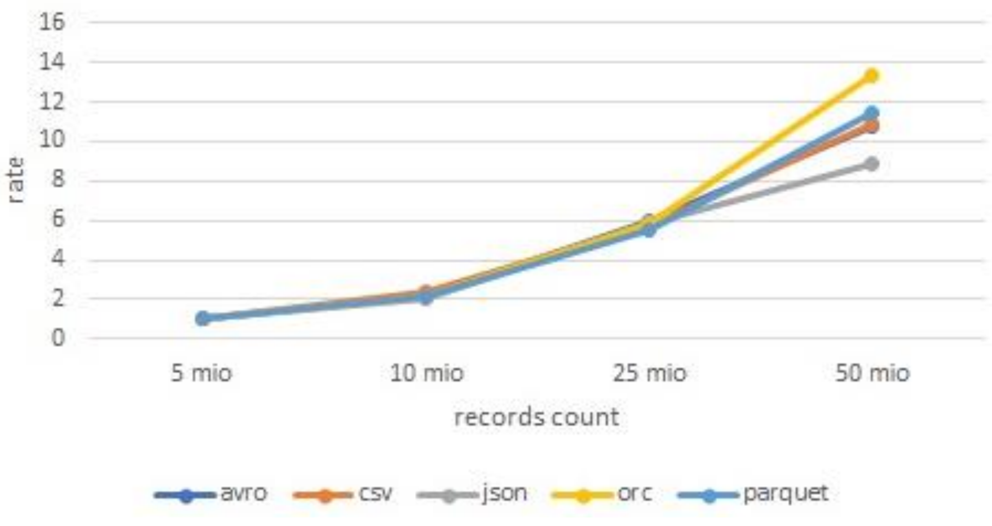

Figure 8. Duration changing by reading all lines.

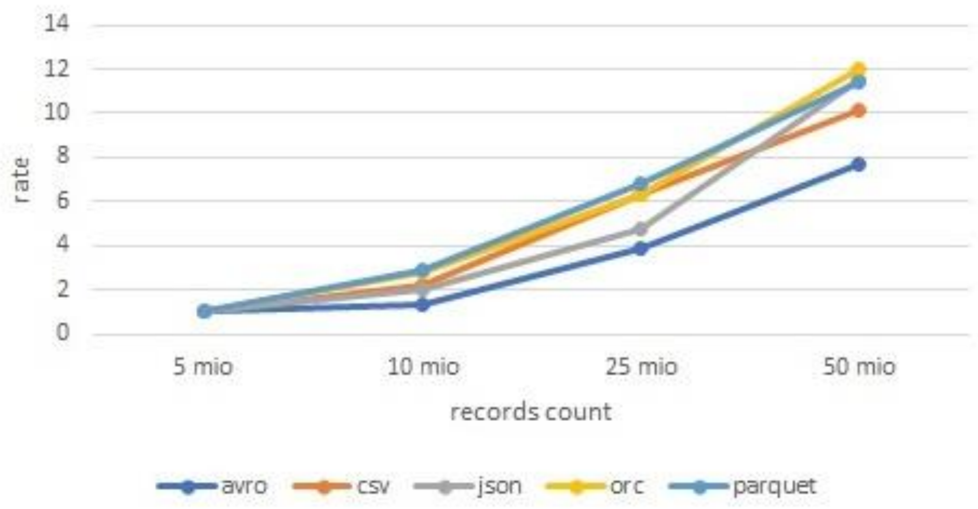

Figure 9. Duration changing by searching for unique objects.

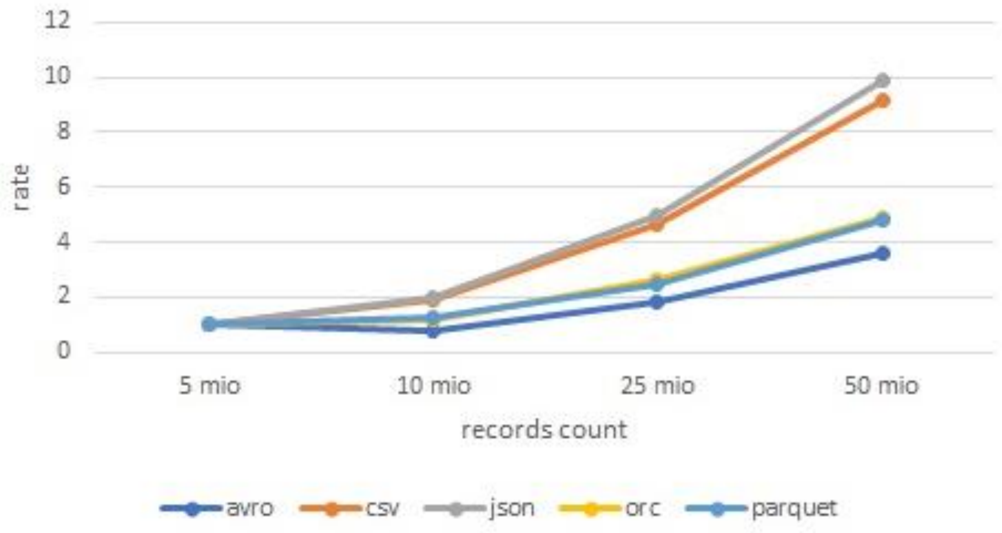

Figure 10. Duration changing by sorting.

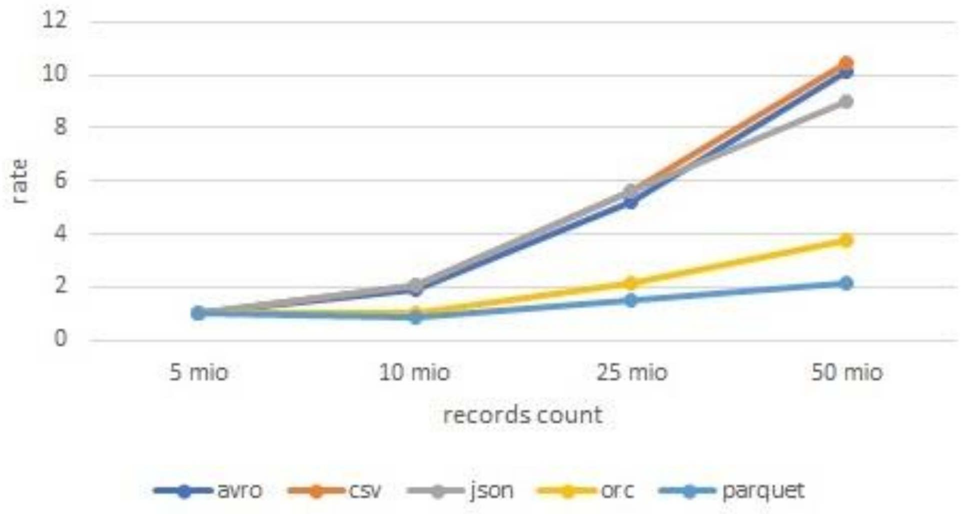

Figure 11. Duration changing by filtering.

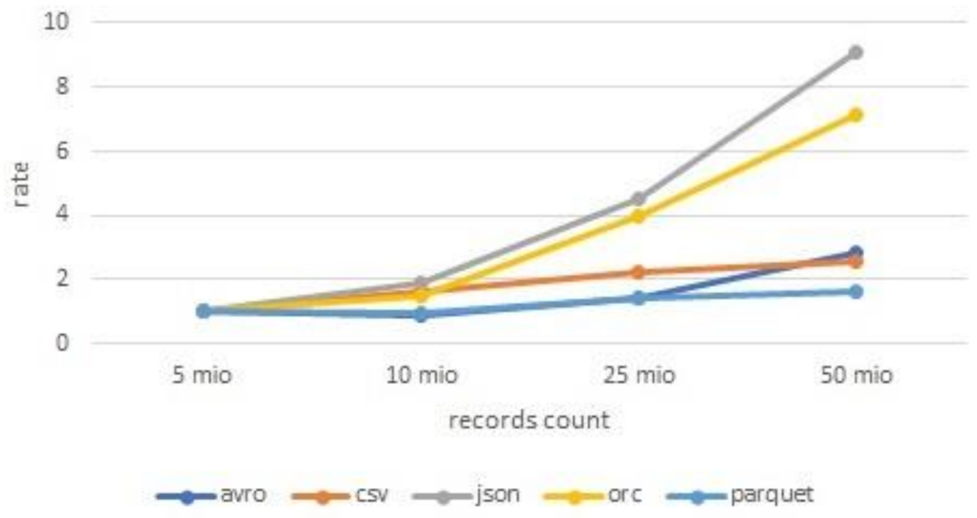

Figure 12. Duration changing by grouping.

As presented, there is an anomaly in the operations of sorting, filtering, and grouping in the form of a slight change in the processing time of the files. This requires studying the algorithm for processing these formats for hidden functions built into the Apache Spark framework that affect such changes.

3.2. *Analysis of the Spark Algorithm*

To further compare the storage formats of big data, let us analyze the algorithms used by the framework for each operation for each data storage format. It should be understood how the framework works with each data storage format. To obtain statistical information on the operation of the algorithm, the Spark Web User Interface tool built into the main framework [32] was used, which collects information about the operation of the application.

As stated earlier, the Spark framework supports lazy evaluation. Therefore, to analyze the algorithms, two operations were performed: transformation, represented by the operation under study, and action, which is the operation of counting the number of objects in the dataset.

The following were chosen as the main metrics:

- Stages count;
- Task count on each stage;
- Shuffle spill (memory/drive) on each stage;
- Median value statistics.

For example, consider the following three operations performed:

- Searching for unique objects;
- Data filtering;
- Sorting.

Search for unique objects. Figure 13 shows an algorithm that is common to all data storage formats. As it can be seen in the figure, the algorithm consists of three stages. Appendix B contains the detailed characteristics obtained for this operation.

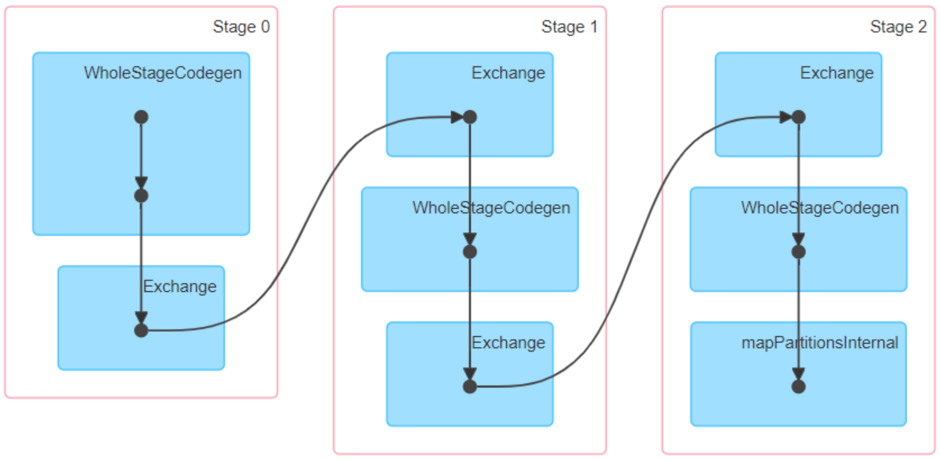

Figure 13. Algorithm schema of the searching for unique objects.

Data filtering. Data filtering algorithm consists of two stages. Figure 14 shows the schema of this algorithm. Appendix B contains the detailed characteristics obtained for this operation. As it can be seen from the figure, the algorithm consists of three stages. Each stage consists of the code function generation (WholeStageCodegen) and shuffle stage (Exchange). Appendix B contains detailed characteristics obtained for this operation.

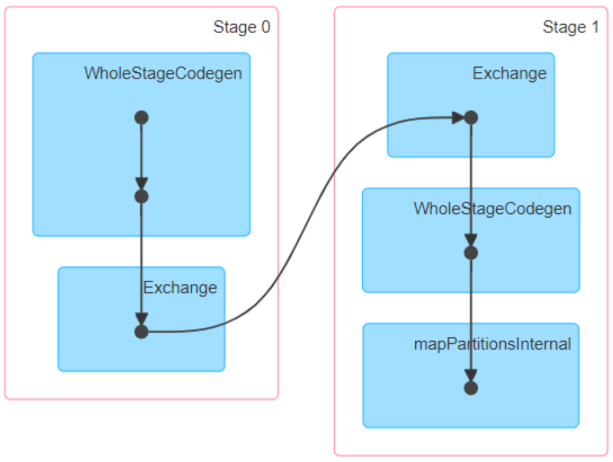

Figure 14. Algorithm schema of data filtering.

Sorting. Sorting, unlike the previous two operations, consists of two jobs, each of which consists of one or more stages. Figures 15 and 16 show the algorithm for working with files on the first and second job, respectively. Appendix B contains detailed characteristics obtained for this operation.

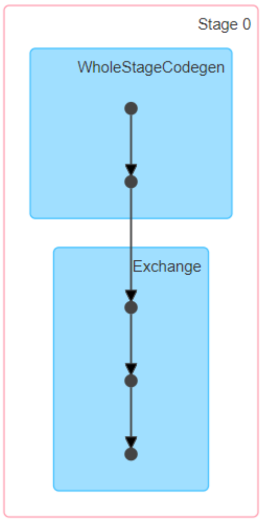

Figure 15. Stage 0 algorithm schema.

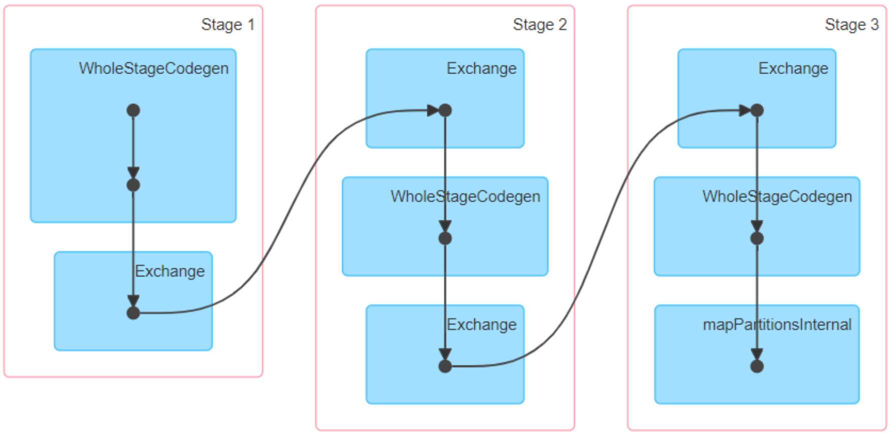

Figure 16. Stages 1–3 algorithm schema.

Following the above analysis, the data processing algorithm is the same for each storage format.

The differences in the results obtained are insignificant, which means that the results obtained during the experiment are typical for the presented data storage formats. Thus, the framework does not significantly affect the conduct of the experiment.

4. Results

The problem of choosing the optimal format was presented in the form of several optimization tasks using the tropical optimization algorithm [33,34].

The aim of the algorithm is to calculate the rating vector of the alternatives presented. It consists of the following stages:

1. Paired comparisons of each alternative;
2. The comparison of the criteria themselves;
3. Optimization task solution.

It should be noted that the comparison of alternatives is based on the required task. Appendix C contains the tables of paired comparison of the criteria and alternatives.

There are no special rules for evaluating alternatives. Each researcher has the right to determine the rules for the comparative assessment of alternatives. In addition, the assessment of alternatives and criteria depends on the tasks assigned to the researcher.

It is important to know the following features of compiling comparison matrices:

- a_{ij} describes the degree of preference for alternative i over alternative j;
- $a_{ij} = a_{ji}^{-1}$.

As part of the current research, the following methodology for evaluating ratings was developed. It consists of the following rules for choosing preferences:

1. Platform independence is not the most important characteristic, because the study is aimed to find the optimal file format for Apache Hadoop system.
2. The ability to record complex structures has an important role, since it provides great opportunities for data processing and analysis.
3. The ability to modify data is not critical, since most big data storage platforms comply with the "write once—read many" principle.
4. The possibility of compression has an indirect role since it affects the volume of data.
5. The presence of metadata is an indicator that does not require analysis, because it affects the speed of reading and grouping data.

According to the experiments results, the following rules were formulated:

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);
- If necessary, it is possible to use intermediate values.

Before describing the algorithm, it is necessary to introduce the basic definitions of tropical algebra [33].

Consider the set of positive real numbers $\mathbb{R}_+$, on which two operations are defined: the operation of idempotent addition $\oplus$ with a neutral element 0, the result of which is the choice of the maximum of the terms, and the operation of multiplication $\otimes$ with a neutral element 1 (defined as usual). For each element x on the set, an inverse element x^{-1} is defined, such that $xx^{-1} = x^{-1}x = 1$. The resulting system is called the idempotent semifield.

The definition of matrices in an idempotent semifield is usual.

The trace is defined as follows:

$$tr\ A = \ a_{11} \oplus a_{nn}$$

The tropical spectral radius of a matrix is a following scalar:

$$\lambda = \oplus_{m=1}^{n}\, tr^{\frac{1}{m}}(A^m)$$

The asterate operator means next operation:

$$A^* = I \oplus A\ \oplus\ \ldots \oplus\ A^{n-1}$$

Using the given matrices, calculate the rating vector of alternatives [34]. The algorithm for calculating the rating vector of alternatives consists of the following steps:

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

$$w = \left(\mu^{-1}C\right)^{*}, \text{ where } \mu = tr\, C \oplus \ldots \oplus tr^{\frac{1}{m}}(C^{m}) \text{ and } C \text{ is criteria matrix}$$

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

$$w_1 = \left(\delta^{-1}\mathbf{1}\mathbf{1}^{T} \oplus \mu^{-1}C\right)^{*}, \text{ where } \delta = \mathbf{1}^{T}\left(\mu^{-1}C\right)^{*}\mathbf{1}$$

and the most differentiating vectors:

$$w_2 = P\left(I \oplus P_{sk}^{-}P\right),$$

where P is a matrix $\left(\mu^{-1}C\right)^{*}$ 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:

$$k = \underset{j}{\operatorname{argmax}}\mathbf{1}^{T}p_j p_j^{-}\mathbf{1},\ s = \underset{i}{\operatorname{argmax}}p_{ik}^{-1}.$$

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

$$D_1 = w_1^{(1)}A_1 \oplus \ldots \oplus\ w_m^{(1)}A_m,\ D_2 = w_1^{(2)}A_1 \oplus \ldots \oplus\ w_m^{(2)}A_m$$

4. Calculate the least differentiating vector of the rating of alternatives:

$$x_1 = \left(v_1^{-1}D_1\right)^{*},\ where\ v_1 = tr\ D_1 \oplus \ldots \oplus tr^{\frac{1}{n}}(D_1^{n})$$

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

$$x_1 = \left(\delta_1^{-1}\mathbf{1}\mathbf{1}^{T} \oplus v_1^{-1}D_1\right)^{*}\ where\ \delta_1 = \mathbf{1}^{T}\left(v_1^{-1}D_1\right)^{*}\mathbf{1}$$

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

$$x_2 = \left(v_2^{-1}D_2\right)^{*},\ where\ v_2 = tr\ D_2 \oplus \ldots \oplus tr^{\frac{1}{n}}(D_2^{n})$$

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

$$x_2 = Q\left(I \oplus Q_{sk}^{-}Q\right)^{*},$$

where Q is a matrix $\left(v_2^{-1}D_2\right)^{*}$ 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:

$$k = arg\ \underset{j}{max}\ \mathbf{1}^{T}q_j q_j^{-}\mathbf{1},\ s = arg\ \underset{i}{max}\ q_{ik}^{-1}.$$

At first, calculate spectral radius using calculation rules in the independent semifield –μ = 1.5874.

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

$$w_1 \approx \begin{pmatrix} 1 & 1.26 & 1.26 & 3.17 & 3.17 & 4 & 4 & 4 & 4 & 1.26 \\ 0.63 & 0.8 & 0.8 & 2 & 2 & 2.5 & 2.5 & 2.5 & 2.5 & 1 \end{pmatrix}^T$$

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:

$$x_1 \approx \begin{pmatrix} 1 & 1 & 0.56 & 1.42 & 1.42 \end{pmatrix}^T$$

Let us calculate the most differentiating vector of rating of alternatives. At first, calculate weights vector for this:

$$w_2 \approx \begin{pmatrix} 1 & 2 & 2 & 5 & 5 & 6.35 & 6.35 & 6.35 & 6.35 & 1.6 \\ 0.4 & 0.8 & 0.8 & 2 & 2 & 2.5 & 2.5 & 2.5 & 2.5 & 1 \end{pmatrix}^T$$

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

$$x_2 \approx \begin{pmatrix} 1 & 1 & 0.56 & 1.42 & 2 \end{pmatrix}^T$$

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

$$parquet \cong orc \succ avro \cong csv \succ json.$$

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

5. Discussion

This study presented is an example of the application of experimental evaluation and tropical optimization methods to find the optimal data storage format when developing a data processing and storage system using the Apache Hadoop platform and the Apache Spark framework.

This study can be used to build data processing and storage systems based on the Apache Hadoop platform or similar solutions. In addition, it can be an example of a solution to similar problems when a selection from a list of alternatives is required. Such questions can arise both when choosing data storage formats and other tools and system components.

The resulting solution is based on the results of specific tests and does not reflect the popularity or functionality of the formats under consideration, it only reflects the expediency of using the formats in the conditions under consideration—the presence of big data and the use of the Apache Hadoop platform.

However, unlike other similar studies [14,17–19], this study solved the problem of choosing an effective solution by methods of tropical algebra based on matrices constructed on the basis of experimental parameter estimates. The use of the proposed approach made it possible to take into account several investigated parameters for evaluating data storage formats without introducing additional hypotheses about the priorities of the evaluation criteria. The use of tropical analysis tools, its symmetric properties during the transition to idempotent semirings made it possible to form an algorithm for choosing solutions, which will expand its use for similar problems when using other formats or other experimental methods.

For example, big data processing systems are cluster systems, which allow processing more data using several nodes connected to a computer network. In this study, a single node was used, the results of which may differ from clustering a similar dataset. Therefore, the authors plan to continue the experiment using clusters with different types of configuration and resources.

In addition, the rate of change in processing time formats depending on the volume should be studied.

6. Conclusions

The paper presents a methodology for choosing a data storage format based on an experimental evaluation of the five most popular formats using the Apache Spark framework.

The study consisted of two parts: experimental and computational. In the experimental part, each format was evaluated based on several test runs. 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 characteristics of data storage formats, such as the volume and processing speed for different operations using the Apache Spark framework.

In the computational part, an algorithm for choosing alternatives using tropical optimization methods was presented, the essence of which is to solve a multi-criteria decision-making problem, which results is presented in the form of vector of preference degrees.

The article also provides an example of assigning ratings for alternatives. The algorithm helps to find the optimal solution for the specific requirements of the system.

The contribution of the study (presented in this paper) is that a technique of choosing a data storage format has been developed, using the example of experimental assessment and the methods of tropical algebra. As an example, the formats supported by the Apache Hadoop system and the Apache Spark framework, as one of the most popular frameworks for processing big data, were used.

It should be noted that this study was not aimed at studying the functional features of the presented data storage formats. The main goal of the study is to build a rating of the presented alternatives based on their experimental assessment using the entered parameters necessary for solving a specific problem.

These techniques can be useful for practical use. In any company, when developing or using software, there is always a choice of which package or system to use. An important selection criterion is the exchange of data with other components, which are often determined by data formats. In the paper, for the considered use cases of big data, the choice of the best solution was made, which can be useful in a similar case. However, the methods and experimental studies and quality indicators, as well as the optimization algorithm, are described in sufficient detail, and can be used for similar tasks where the choice of format is important, while the conditions for using the formats and the set of alternative options may be different.

This study can be one example for solving similar problems, without introducing additional hypotheses concerning the priorities of the evaluation criteria.

Author Contributions: Conceptualization, E.N. and A.T.; methodology, V.B.; software, V.B.; validation, E.N., V.B.; formal analysis, A.T. and V.B.; data curation, V.B.; writing—original draft preparation, V.B. and E.N.; writing—review and editing, A.T.; visualization, V.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Higher Education of the Russian Federation.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A Experimental Resources

To evaluate experimental assessment, the experimental stand was built. The stand configuration is presented in Table A1. Table A2 contains the description of the data generated for the study.

Table A1. 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 A2. 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

Appendix B Statistics of the Operations Performed

The statistics obtained during the experimental evaluation are presented below.

Tables A3–A5 show the comparative characteristics at each stage for operation of searching for unique objects.

Table A3. Stage 0 characteristics of the searching for unique values.

Criteria	Avro	CSV	JSON	ORC	Parquet
Task count	9	10	20	9	9
Shuffle spill (memory/disk)	0.0 B/ 703.0 MB	0.0 B/ 611.5 MB	-	0.0 B/ 702.9 MB	0.0 B/ 703.2 MB
Median Values Statistics by Task					
Scheduler Delay	20 ms	11 ms	6 ms	15 ms	9 ms
Task Deserialization Time	29 ms	21 ms	3 ms	30 ms	17 ms
Garbage Collection Time	2 s	2 s	0.3 s	2 s	3 s
Peak Execution Memory	232.0 MB	248.4 MB	144.0 MB	232.0 MB	232.0 MB
Shuffle spill (memory/disk)	0.0 B/ 85.5 MB	0.0 B/ 76.2 MB	-	0.0 B/ 85.5 MB	0.0 B/ 85.5 MB

Table A4. Stage 1 characteristics of the searching for unique values.

Criteria	Avro	CSV	JSON	ORC	Parquet
Task count	200	200	200	200	200
Shuffle write	11.5 KB/200	11.5 KB/200	11.5 KB/200	11.5 KB/200	11.5 KB/200
Median Values Statistics by Task					
Scheduler Delay	1 ms	2 ms	1 ms	1 ms	2 ms
Task Deserialization Time	1 ms	2 ms	2 ms	1 ms	1 ms
Garbage Collection Time	0 ms	0 ms	0 ms	0 ms	0 ms
Peak Execution Memory	18.0 MB	18.0 MB	18.0 MB	18.0 MB	18.0 MB
Shuffle Read Size/Records	4.0 MB/50,018	4.5 MB/50,013	4.1 MB/50,018	4.0 MB/50,018	4.0 MB/50,018
Shuffle Write Size/Records	59.0 B/1	59.0 B/1	59.0 B/1	59.0 B/1	59.0 B/1

Table A5. Stage 2 characteristics of the searching for unique values.

Criteria	Avro	CSV	JSON	ORC	Parquet
Task count	1	1	1	1	1
Median Values Statistics by Task					
Scheduler Delay	2 ms	0 ms	0 ms	0 ms	1 ms
Task Deserialization Time	1 ms	2 ms	1 ms	2 ms	0 ms
Garbage Collection Time	0 ms	0 ms	0 ms	0 ms	0 ms
Peak Execution Memory	0.0 B	0.0 B	0.0 B	0.0 B	0.0 B

Below are Tables A6 and A7, showing the values of the main characteristics obtained at each stage for operation of the data filtering.

Table A6. Stage 0 characteristics of the sorting.

Criteria	Avro	CSV	JSON	ORC	Parquet
Task count	9	10	20	9	9
Median Values Statistics by Task					
Scheduler Delay	52 ms	48 ms	17 ms	0.1 s	28 ms
Task Deserialization Time	26 ms	20 ms	4 ms	32 ms	10 ms
Garbage Collection Time	97 ms	93 ms	17 ms	45 ms	43 ms
Peak Execution Memory	0.0 B	0.0 B	0.0 B	0.0 B	0.0 B

Table A7. Stage 1 characteristics of the sorting.

Criteria	Avro	CSV	JSON	ORC	Parquet
Task count	9	10	20	9	9
Median Values Statistics by Task					
Scheduler Delay	4 ms	3 ms	2 ms	3 ms	2 ms
Task Deserialization Time	17 ms	1 ms	7 ms	11 ms	18 ms
Garbage Collection Time	0.2 s	0.2 s	86 ms	0.2 s	0.1 s
Peak Execution Memory	0.0 B	0.0 B	0.0 B	0.0 B	0.0 B
Shuffle Write Size/Records	5.5 MB/1,166,764	4.8 MB/1,000,000	2.8 MB/582,729	5.5 MB/1,166,764	5.6 MB/1,166,764

Tables A8–A11 describe each stage of the sorting operation.

Table A8. Stage 0 characteristics of the sorting.

Criteria	Avro	CSV	JSON	ORC	Parquet
Task count	200	200	200	200	200
Shuffle write	11.5 KB/200	11.5 KB/200	11.5 KB/200	11.5 KB/200	11.5 KB/200
Median Values Statistics by Task					
Scheduler Delay	2 ms	1 ms	2 ms	1 ms	1 ms
Task Deserialization Time	2 ms	2 ms	2 ms	1 ms	2 ms
Garbage Collection Time	0 ms	0 ms	0 ms	0 ms	0 ms
Peak Execution Memory	10.0 MB	10.0 MB	10.0 MB	10.0 MB	10.0 MB
Shuffle Read Size/Records	242.7 KB/50,856	243.9 KB/50,955	243.3 KB/50,943	242.5 KB/50,934	243.8 KB/50,908
Shuffle Write Size/Records	59.0 B/1	59.0 B/1	59.0 B/1	59.0 B/1	59.0 B/1

Table A9. Stage 1 characteristics of the sorting.

Criteria	Avro	CSV	JSON	ORC	Parquet
Task count	1	1	1	1	1
Median Values Statistics by Task					
Scheduler Delay	1 ms	2 ms	1 ms	0 ms	1 ms
Task Deserialization Time	1 ms	1 ms	1 ms	1 ms	1 ms
Garbage Collection Time	0 ms	0 ms	0 ms	0 ms	0 ms
Peak Execution Memory	0.0 B	0.0 B	0.0 B	0.0 B	0.0 B

Table A10. Stage 2 characteristics of the sorting.

Criteria	Avro	CSV	JSON	ORC	Parquet
Task count	200	200	200	200	200
Shuffle write	11.5 KB/200	11.5 KB/200	11.5 KB/200	11.5 KB/200	11.5 KB/200
Median Values Statistics by Task					
Scheduler Delay	2 ms	1 ms	2 ms	1 ms	1 ms
Task Deserialization Time	2 ms	2 ms	2 ms	1 ms	2 ms
Garbage Collection Time	0 ms	0 ms	0 ms	0 ms	0 ms
Peak Execution Memory	10.0 MB	10.0 MB	10.0 MB	10.0 MB	10.0 MB
Shuffle Read Size/Records	242.7 KB/50,856	243.9 KB/50,955	243.3 KB/50,943	242.5 KB/50,934	243.8 KB/50,908
Shuffle Write Size/Records	59.0 B/1	59.0 B/1	59.0 B/1	59.0 B/1	59.0 B/1

Table A11. Stage 3 characteristics of the sorting.

Criteria	Avro	CSV	JSON	ORC	Parquet
Task count	1	1	1	1	1
Median Values Statistics by Task					
Scheduler Delay	1 ms	2 ms	1 ms	0 ms	1 ms
Task Deserialization Time	1 ms	1 ms	1 ms	1 ms	1 ms
Garbage Collection Time	0 ms	0 ms	0 ms	0 ms	0 ms
Peak Execution Memory	0.0 B	0.0 B	0.0 B	0.0 B	0.0 B

Appendix C Matrices of Alternatives Comparisons

To assess the alternatives, matrices of comparison of criteria and alternatives were compiled for each criterion.

Table A12 describes the criteria preference matrix.

Table A12. Critria preference matrix.

Criteria	Platform Independence	Recording Complex Structure	Volume	Reading All Lines	Filter	Unique Values	Grouping	Sorting
Platform independence	1	1/2	1/2	1/5	1/5	1/4	1/4	1/2
Recording complex structure	2	1	1	1/4	1/4	1/2	1/2	1
Volume	2	1	1	1/4	1/4	1/2	1/2	1
Reading all lines	5	4	4	1	1	1/2	1/2	2
Filter	5	4	4	1	1	1/2	1/2	2
Unique values	4	2	2	2	2	1	1	4
Grouping	4	2	2	2	2	1	1	4
Sorting	2	1	1	1/2	1/2	1/4	1/4	1

Tables A13–A20 describe the matrices of the alternatives according to each criterion.

Table A13. Rating of alternatives according to platform independence.

	Avro	CSV	JSON	ORC	Parquet
avro	1	1	1	3	3
csv	1	1	1	3	3
json	1	1	1	3	3
orc	1/3	1/3	1/3	1	1
parquet	1/3	1/3	1/3	1	1

Table A14. Rating of alternatives according to recording complex structure.

	Avro	CSV	JSON	ORC	Parquet
avro	1	3	1	1	1
csv	1/3	1	1/3	1/3	1/3
json	1	3	1	1	1
orc	1	3	1	1	1
parquet	1	3	1	1	1

Table A15. Rating of alternatives according to volume.

	Avro	CSV	JSON	ORC	Parquet
avro	1	2	4	1/2	1/2
csv	1/2	1	3	1/3	1/3
json	1/4	1/3	1	1/4	1/4
orc	2	3	4	1	1
parquet	2	3	4	1	1

Table A16. Rating of alternatives according to reading all lines.

	Avro	CSV	JSON	ORC	Parquet
avro	1	1	3/2	3/4	1/2
csv	1	1	3/2	3/4	1/2
json	2/3	2/3	1	1/2	1/3
orc	4/3	4/3	2	1	2/3
parquet	2	2	3	3/2	1

Table A17. Rating of alternatives according to filter.

	Avro	CSV	JSON	ORC	Parquet
avro	1	1	2	1/4	1/4
csv	1	1	1/2	1/4	1/4
json	1/2	1/2	1	1/4	1/4
orc	4	4	4	1	1
parquet	4	4	4	1	1

Table A18. Rating of alternatives according to searching for unique values.

	Avro	CSV	JSON	ORC	Parquet
avro	1	8/7	8/7	1	5/7
csv	7/8	1	1	7/8	5/8
json	7/8	1	1	7/8	5/8
orc	1	8/7	8/7	1	5/7
parquet	7/5	8/5	8/5	7/5	1

Table A19. Rating of alternatives according to grouping.

	Avro	CSV	JSON	ORC	Parquet
avro	1	1/2	2	1/2	1/3
csv	2	1	3	4/5	3/5
json	1/2	1/3	1	1/4	1/5
orc	2	5/4	4	1	3/4
parquet	3	5/3	5	4/3	1

Table A20. Rating of alternatives according to sorting.

	Avro	CSV	JSON	ORC	Parquet
avro	1	1	3/2	2/3	1/2
csv	1	1	3/2	2/3	1/2
json	2/3	2/3	1	1/2	1/3
orc	3/2	3/2	2	1	2/3
parquet	2	2	3	3/2	1

References

1. Chong, D.; Shi, H. Big data analytics: A literature review. *J. Manag. Anal.* **2015**, *2*, 175–201. [CrossRef]
2. Moro Visconti, R.; Morea, D. Big Data for the Sustainability of Healthcare Project Financing. *Sustainability* **2019**, *11*, 3748. [CrossRef]
3. Ardito, L.; Scuotto, V.; Del Giudice, M.; Messeni, A. A bibliometric analysis of research on Big Data analytics for business and management. *Manag. Decis.* **2018**, *57*, 1993–2009. [CrossRef]
4. Cappa, F.; Oriani, R.; Peruffo, E.; McCarthy, I.P. Big Data for Creating and Capturing Value in the Digitalized Environment: Unpacking the Effects of Volume, Variety and Veracity on Firm Performance. *J. Prod. Innov. Manag.* **2020**. [CrossRef]
5. Yang, C.; Huang, Q.; Li, Z.; Liu, K.; Hu, F. Big Data and cloud computing: Innovation opportunities and challenges. *Int. J. Digit. Earth* **2017**, *10*, 13–53. [CrossRef]
6. Mavridis, I.; Karatza, H. Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark. *J. Syst. Softw.* **2017**, *125*, 133–151. [CrossRef]
7. Lee, S.; Jo, J.Y.; Kim, Y. Survey of Data Locality in Apache Hadoop. In Proceedings of the 2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD), Honolulu, HI, USA, 29–31 May 2019; pp. 46–53.
8. Garg, K.; Kaur, D. Sentiment Analysis on Twitter Data using Apache Hadoop and Performance Evaluation on Hadoop MapReduce and Apache Spark. In Proceedings of the International Conference on Artificial Intelligence (ICAI), Las Vegas, NV, USA, 29 July–1 August 2019; pp. 233–238.
9. Hive. 2020 Apache Hive Specification. Available online: https://cwiki.apache.org/confluence/display/HIVE (accessed on 11 January 2021).

10. Impala. 2020 Apache Impala Specification. Available online: https://impala.apache.org/impala-docs.html (accessed on 11 January 2021).
11. Nazari, E.; Shahriari, M.H.; Tabesh, H. BigData Analysis in Healthcare: Apache Hadoop, Apache spark and Apache Flink. *Front. Health Inform.* **2019**, *8*, 14. [CrossRef]
12. Salloum, S.; Dautov, R.; Chen, X.; Peng, P.X.; Huang, J.Z. Big data analytics on Apache Spark. *Int. J. Data Sci. Anal.* **2016**, *1*, 145–164. [CrossRef]
13. Krivulin, N. A new algebraic solution to multidimensional minimax location problems with Chebyshev distance. *WSEAS Trans. Math.* **2012**, *11*, 605–614.
14. Gusev, A.; Ilin, D.; Nikulchev, E. The Dataset of the Experimental Evaluation of Software Components for Application Design Selection Directed by the Artificial Bee Colony Algorithm. *Data* **2020**, *5*, 59. [CrossRef]
15. Ramírez, A.; Parejo, J.A.; Romero, J.R.; Segura, S.; Ruiz-Cortés, A. Evolutionary composition of QoS-aware web services: A many-objective perspective. *Expert Syst. Appl.* **2017**, *72*, 357–370. [CrossRef]
16. Gholamshahi, S.; Hasheminejad, S.M.H. Software component identification and selection: A research review. *Softw. Pract. Exp.* **2019**, *49*, 40–69. [CrossRef]
17. Gusev, A.; Ilin, D.; Kolyasnikov, P.; Nikulchev, E. Effective Selection of Software Components Based on Experimental Evaluations of Quality of Operation. *Eng. Lett.* **2020**, *28*, 420–427.
18. Kudzh, S.A.; Tsvetkov, V.Y.; Rogov, I.E. Life cycle support software components. *Russ. Technol. J.* **2020**, *8*, 19–33. [CrossRef]
19. Munir, R.F.; Abelló, A.; Romero, O.; Thiele, M.; Lehner, W. A cost-based storage format selector for materialized results in big data frameworks. *Distrib. Parallel Databases* **2020**, *38*, 335–364. [CrossRef]
20. Nicholls, B.; Adangwa, M.; Estes, R.; Iradukunda, H.N.; Zhang, Q.; Zhu, T. Benchmarking Resource Usage of Underlying Datatypes of Apache Spark. *arXiv* **2020**, arXiv:2012.04192. Available online: https://arxiv.org/abs/2012.04192 (accessed on 11 January 2021).
21. Wang, X.; Xie, Z. The Case for Alternative Web Archival Formats to Expedite The Data-To-Insight Cycle. *arXiv* **2020**, arXiv:2003.14046.
22. He, D.; Wu, D.; Huang, R.; Marchionini, G.; Hansen, P.; Cunningham, S.J. Proceedings of the ACM/IEEE Joint Conference on Digital Libraries 2020 in Wuhan virtually. *ACM Sigweb Newsl.* **2020**, *1*, 1–7. [CrossRef]
23. Ahmed, S.; Ali, M.U.; Ferzund, J.; Sarwar, M.A.; Rehman, A.; Mehmood, A. Modern Data Formats for Big Bioinformatics Data Analytics. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*. [CrossRef]
24. Plase, D.; Niedrite, L.; Taranovs, R. A Comparison of HDFS Compact Data Formats: Avro Versus Parquet. *Moksl. Liet. Ateitis* **2017**, *9*, 267–276. [CrossRef]
25. Khan, S.; Liu, X.; Ali, S.A.; Alam, M. Storage Solutions for Big Data Systems: A Qualitative Study and Comparison. *arXiv* **2019**, arXiv:1904.11498. Available online: https://arxiv.org/abs/1904.11498 (accessed on 11 January 2021).
26. Moniruzzaman, A.B.M.; Hossain, S.A. NoSQL Database: New Era of Databases for Big data Analytics-Classification, Characteristics and Comparison. *Int. J. Database Theory Appl.* **2013**, *6*, 1–14.
27. Apache. Avro specification 2012. Available online: http://avro.apache.org/docs/current/spec.html (accessed on 11 January 2021).
28. ORC. ORC Specification 2020. Available online: https://orc.apache.org/specification/ORCv1/ (accessed on 11 January 2021).
29. Sakr, S.; Liu, A.; Fayoumi, A.G. The family of mapreduce and large-scale data processing systems. *ACM Comput. Surv. (CSUR)* **2013**, *46*, 1–44. [CrossRef]
30. Apache. Parquet Official Documentation 2018. Available online: https://parquet.apache.org/documen-tation/latest/ (accessed on 11 January 2021).
31. Chellappan, S.; Ganesan, D. Introduction to Apache Spark and Spark Core. In *Practical Apache Spark*; Apress: Berkeley, CA, USA, 2018; pp. 79–113.
32. Zaharia, M.; Chowdhury, M.; Franklin, M.J.; Shenker, S.; Stoica, I. Spark: Cluster computing with working sets. *HotCloud* **2010**, *10*, 95.
33. Krivulin, N.; Sergeev, S. Tropical optimization techniques in multi-criteria decision making with Analytical Hierarchy Process. In Proceedings of the 2017 European Modelling Symposium (EMS), Manchester, UK, 20–21 November 2017; pp. 38–43.
34. Krivulin, N. Methods of tropical optimization in rating alternatives based on pairwise comparisons. In *Operations Research Proceedings 2016*; Springer: Cham, Germany, 2018; pp. 85–91.