MOKSLAS – LIETUVOS ATEITIS
SCIENCE – FUTURE OF LITHUANIA
ISSN 2029-2341 / eISSN 2029-2252
http://www.mla.vgtu.lt

ELEKTRONIKA IR ELEKTROTECHNIKA
ELECTRONICS AND ELECTRICAL ENGINEERING

2017 9(3): 267–276

https://doi.org/10.3846/mla.2017.1033

A COMPARISON OF HDFS COMPACT DATA FORMATS: AVRO VERSUS PARQUET

Daiga PLASE[1], Laila NIEDRITE[2], Romans TARANOVS[3]
[1,2]University of Latvia, Riga, Latvia
[3]Riga Technical University, Riga, Latvia

E-mails: [1]Daiga.Plase@accenture.com; [2]Laila.Niedrite@lu.lv; [3]Romans.Taranovs@accenture.com

Abstract. In this paper, file formats like Avro and Parquet are compared with text formats to evaluate the performance of the data queries. Different data query patterns have been evaluated. Cloudera's open-source Apache Hadoop distribution CDH 5.4 has been chosen for the experiments presented in this article. The results show that compact data formats (Avro and Parquet) take up less storage space when compared with plain text data formats because of binary data format and compression advantage. Furthermore, data queries from the column based data format Parquet are faster when compared with text data formats and Avro.

Keywords: Big Data, Hadoop, HDFS, Hive, Avro, Parquet.

Introduction

The amount of data captured by social media, the Internet of Things, enterprises and different types of applications is growing exponentially. Every day people leave an incredible amount of data behind them in the digital environment. It is not without a reason that data are called "the new oil" nowadays. If data are used skillfully, companies can increase their revenues, predict future prospects and go ahead of the competition. There are huge volumes of raw data every day. However, these data do not yield much information until processed. Because of processing, raw data sometimes end up in a database, which enables the data to become accessible for further processing and analysis in a number of different ways.

Towards distributed and real-time processing of large data sets – so-called Big Data – the traditional computing techniques are becoming insufficient (Chandra *et al.* 2012; Grover *et al.* 2015; Sharma *et al.* 2014; Wonjin *et al.* 2014). Hadoop is one of the most common open source Big Data frameworks in the industry today, capable to carry out common Big Data related tasks. There is growing business demand for Hadoop technology usage in Big Data analysis like storage, biological data, road, traffic, travel and tourism, telecommunication, enterprise data, citizen's info (Grover *et al.* 2015; Sharma *et al.* 2014). In addition, Hadoop technology is becoming popular in such areas as cloud computing, internet data management (storage, load balancing), implementing MapReduce algorithms for providing solutions to various problems of handling large amount of data, in proposing new models by using HDFS (Sharma *et al.* 2014).

Big data enables organizations to gather, store, and manipulate vast amounts of data at the right speed and time. Considering big data advantages, many companies are starting to leverage big data and advanced analytics to increase their market share. In order to maintain and improve on its market position, companies need to leverage advanced analytics to better inform its marketing, sales and operation functions through effective customer profiling and insights.

The experience of the authors shows, that there is a growing business interest on how to store data better in Hadoop and which data format usage provides faster access to data with different kind of queries, e.g. scan and aggregate queries. Some requirements for data and analytics platform cover the need to store everything, analyze anything, and build what users need to answer a full range of questions from simple ones: "what happened", "how many", "how often", "in what place", "where is the problem" to advanced ones: "what is happening now", "what will happen if this trend continues", and "what is the best option".

There is some amount of untapped value in the Big Data. Data come from satellite images, eCommerce, TV, GPS, video sensors, social media, the Internet of Things, enterprises and different type of applications, from variety of sources. It is necessary to correlate all data, use analytics and predictive analytics, deep analytics, deeper insight of data in order to come up with answers, to improve the return of investment, to predict extreme weather conditions etc. This need is important regardless of the purpose like just looking at biodiversity trends or trying to understand customers, learn their habits and predict their future behaviors.

Often raw data is stored in specific text formats, for instance: JSON, CSV, XML, etc. These formats allow data to be structured and available for humans to read and edit it in the most convenient manner. However, storing raw data in a plain text has a significant drawback – there is a disk space need to store such files. However, for Big Data cluster powered by Hadoop it is even a bigger problem because of the high replication factor of each data block within Hadoop File System – HDFS.

For instance, recommended HDFS replication factor is 3. That means each raw data block will be replicated 3 times across data nodes. Thus, it is crucial to select appropriate data format that enables HDFS storage space utilization in a more efficient manner according to the task defined. Secondly, data storage format may influence the speed of data processing with Hadoop tools, like Hive. Several binary data storage formats exist. Some of them are RCFile, ORC, Avro, Parquet. These formats were designed for systems that use MapReduce kinds of framework. A structure is a systematic combination of multiple components including data storage format, data compression, and optimization techniques for data reading.

There is another application area of binary data storage format utilization on direct data sources. For instance, service data gathering from mobile phones to get specific insights of people's behavior or in order to create another kind of location intelligence reports. Assuming that a GPS data packet (timestamp, longitude, latitude) is 100 B in average and that smartphone generates it every 8 s, quick math calculations result in 0.043 MB/h, 1.03 MB/day and 376 MB/year. In 2014, over 1.2 billion smartphones were sold (Gartner 2014). If 1 billion devices produce a GPS data packet every 8 s, it results in 1 PB/day. This means that we need ~1000 disk drives with size 1 TB in order to store these data. The volume of data is enormous. The question is where and how to store this data in order to provide a database for faster execution of data queries. This is the main rationale for this article.

This article is based on the previously carried out systematic literature review of the research direction in Big Data projects using Hadoop Technology, MapReduce kind of framework and compact data formats such as Avro and Parquet (Plase 2016). An experimental investigation was performed.

The rest of the paper is organized as follows: in section "Background" the current status of the research question has been analyzed and background information on the main research topics and terms are given. Section "Goals and objectives" describes the research problem, goals, research questions and hypotheses. Section "Research methodology" presents the research methodology, experimental environment and how the experiments have been performed. Section "Results" comprises the result set and interpretation, followed by conclusions in the last section.

Background

The Hadoop Technology is commonly being used to manage Big Data projects. Hadoop is now the de facto standard for storing and processing big data, not only for unstructured data but also for some structured data (Chen *et al.* 2014). The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications (Shvachko *et al.* 2010). As a result, providing SQL analysis functionality to the big data resided in HDFS becomes more and more important. Although there are other SQL-on-Hadoop systems such as HortonWorks Stinger or Cloudera Impala, Hive is a pioneer system that supports SQL-like analysis to the data in HDFS (Wonjin *et al.* 2014). Hive has been chosen for the experiments because of the same reason that is mentioned in Section "Research methodology" for Cloudera choice.

The data storage formats mentioned in "Introduction" section has some advantages and disadvantages. As shown in Table 1, only Avro and Parquet data format support both important advantages: schema evolution and compression.

Table 1. Comparison of data file formats

File format	Schema integration	Compression support
Text/CSV (Shafranovich 2005)	–	–
JSON (Bray 2014)	+	–
Avro (Apache 2009a)	+	+
SequenceFile (Apache 2009b)	–	+
RCFile (He *et al.* 2011)	–	+
ORC file (Apache 2017)	–	+
Parquet (Apache 2013)	+	+

Avro (Apache 2009a) is a row-based storage format, also described as a data serialization system similar to Java Serialization. Avro provides rich data structures, a compact, fast, binary data format, a container file to store persistent data, remote procedure call (RPC) features. There is not required code generation to read or write data files nor to use or implement RPC protocols. Alternative systems include Java Serialization, Thrift (Apache 2009c) and Protocol Buffers (Google 2001) that only work with compile time code generation. Furthermore, Avro can provide more optimized runtime performance (Palmer *et al.* 2011).

Avro relies on schemas. A schema defines the structure of the data and is used in data reading and writing process. A data schema is defined with JSON and stored into Avro file during data writing process. When Avro data is read, the schema used when writing it is always present. This allows writing data with no per-value over-heads.

Avro is used to save many small files in a single Avro file in HDFS to reduce the namenode memory usage because of user-defined patterns and specific data encoded into binary sequence and stored into a large containing file (Zhang *et al.* 2014).

Parquet (Apache 2013) is a column-based storage format, optimized for work with multi column datasets. Parquet use cases typically involve working with a subset of those columns rather than entire records. One of the most-often cited advantages of columnar data organizations is data compression (Stonebraker *et al.* 2005) and reduced disk I/O (Abadi *et al.* 2009) that improves performance of analytical queries (Floratou *et al.* 2014). Data compression algorithms perform better on data with low information entropy (high data value locality). Thus, the system achieves the I/O performance benefits of compression without increase of CPU load during the decompression (Abadi *et al.* 2009). The layout of Parquet data files is optimized for queries that process large volumes of data.

There is a business demand to define how to utilize Avro or Parquet and find the best practices. The main question is what the differences in performance (query execution time) between Parquet and Avro are?

Several research papers have been published on both comparison of Hadoop high-level processing tools and languages operating with data in binary formats and their utilization.

Cejka *et al.* (2015) from Siemens AG Company compared the file size of four different formats: Java, Protocol Buffers, Thrift and Avro. Avro's results showed that it is much slower in writing speed, however much faster in reading speed than Protocol Buffers and Thrift. The file compression of Apache Avro is best. In order to evaluate the time of retrieval of entries, the author's defined benchmark was used to retrieve data from such databases as Storacle, H2, MongoDB. However, Parquet format was not analyzed in that paper.

Luckow *et al.* (2015) compared different queries derived from TPC-DS and TPC-HS benchmarks and executed on Hive/Text, Hive/ORC, Hive/Parquet, Spark/ORC, Spark/Parquet. Hive/Parquet showed better execution time than Spark/Parquet. Select, aggregate and join queries were executed on a comparable infrastructure Hive/Spark versus RDBMS. Generally, the RDBMS can outperform Hive and Spark – however, both deliver a solid performance at a lower cost. Avro format was not analyzed there.

Zhang Shuo *et al.* (2014) compared raw data storage formats versus Avro and proposed original solution to store, read and write different small files on HDFS. However, there is no direct comparison of different data formats and Parquet was not presented there. It is worth mentioning that authors selected Avro as a target binary data format and demonstrated its efficiency in both read and write operations.

Grover *et al.* (2015) focused on benchmarking multiple SQL-like big data technologies over Hadoop based distributed file system (HDFS) for Study Data Tabulation Model (SDTM) used in clinical trial databases for improving the efficiency of research in clinical trials. The benchmark proposed in that paper provides an overview of the capabilities of SQL-on-Hadoop platforms such as Hive, Presto, Drill and Spark. The authors mentioned Avro and Parquet formats, but they did not analyze these formats in any kind of comparison. Only Parquet format was mentioned in the future work section as a lightweight and fast format with columnar layout, hence they can significantly boost IO performance.

Floratou *et al.* (2014) compared three analytical job execution environments available in Hadoop ecosystem. Hive on MapReduce, Hive on Tez and Impala have been analyzed here by using a world-renowned benchmark like TPC-H. As a result, the authors confirmed that Impala had better performance versus Hive (both versions). Although, the authors mentioned Parquet and Avro, they did not analyze those formats in any kind of comparison.

Tapiador *et al.* (2014) compared the data set size for different compression and format approaches like CSV(Row), Plain(Row), Snappy(Row), GBIN(Row), Snappy(Column), GBIN (Column). Google Snappy codec gave a much better result as the decompression was faster than that of Deflate (GBIN). It took half of the time to process the histograms (50%) and the extra size occupied on disk was only around 23%. This confirmed the suitability of Snappy codec for data to be stored in HDFS and later on analyzed by Hadoop MapReduce workflows. Although this article gave the answer to the question about compactness, it did not compare Avro versus Parquet in another kind of comparison, for instance, SQL query execution time. The data storage model approaching performance comparison did not give a transparent view of how it was obtained.

There remains a significant gap and need for additional experiments and studies in order to answer the research question about the best practice for data storage in Avro or Parquet format.

Goals and objectives

In context of the information given in “Introduction” and “Background” sections of this article, it is crucial to select an appropriate data format that reduces HDFS storage space and improves the speed of data processing with Hadoop tools, like Hive. The objective of this work is to perform experiments in order to answer the research questions:

- RQ.1: What are the differences in performance (query execution time) between Avro and Parquet?
- RQ.2: Which data format (Avro or Parquet) is more compact?

In order to answer the research questions, the experimental investigation has been chosen as a research method. The experimentation process consisted of five stages. It started with scoping and continued with planning, operation, analysis and interpretation, report. In order to formulate the scope of the experiments, independent variables has been defined. The data format type (Avro / Parquet) has been defined as an independent variable, but performance and compactness – as another. Therefore, the scope of the experiments has been formulated as follows: Analyze data format Avro versus Parquet for the purpose of evaluation with respect to performance and compactness from the point of view of the researcher in the context of a Big Data storage format.

Avro and Parquet choice for the experiments was based on assumption that the row-oriented data access supported by Avro should provide a better performance on scan queries, e.g. when all columns are as interest of the processing, but Parquet format as a counterpart should provide a better performance on column-oriented queries, e.g. when only specific set of those is selected. Thus, the research problem can be expressed as null hypotheses.

H_0^{A} Data format Avro is better than Parquet in performance on scan queries.

H_0^{B} Data format Parquet is better than Avro in performance on aggregation queries.

H_0^{C} There is no difference in the compactness between data format Avro and Parquet.

Each hypothesis H_0^X, where X refers to a certain quantity (A – performance on scan queries, B – performance on aggregation queries, C – compactness) has been measured by the corresponding random variable A^X and P^X – respectively Avro and Parquet data format. For instance, H_0^{C} tests the compactness of the data format Avro A^{C} and Parquet P^{C}. Therefore, the null hypothesis H_0^{C} is expressible as:

$$H_0^{\mathrm{C}} : \boldsymbol{p}\left(A^{\mathrm{C}} > P^{\mathrm{C}}\right) = \boldsymbol{p}\left(P^{\mathrm{C}} > A^{\mathrm{C}}\right), \qquad (1)$$

that is, the probability $\boldsymbol{p}$ that Avro is more compact than Parquet equals the probability that Parquet is more compact than Avro. Correspondingly, the alternative hypothesis H_1^{C} is that there is a difference in probability:

$$H_1^{\mathrm{C}} : \boldsymbol{p}\left(A^{\mathrm{C}} > P^{\mathrm{C}}\right) \neq \boldsymbol{p}\left(P^{\mathrm{C}} > A^{\mathrm{C}}\right). \qquad (2)$$

Research methodology

A. Cluster configuration

Nowadays there exist many different big data management systems, like Oracle’s Big Data Appliance, IBM’s Apache Hadoop, Cloudera’s CDH, Hortonwork’s HDP, Microsoft’s Dryad, Apache Spark, etc. All these systems are mainly focused on big data storage and processing, however they may differ in approaches. For instance, MapReduce idea of processing differs from Spark’s DAG approach. In the current paper, Cloudera Enterprise 5.4 distribution of Hadoop has been selected. The main reason for that is high popularity of the platform because of its openness. Cloudera has incorporated more open source Hadoop ecosystem’s projects than any other platform. Thus, it leads to bigger popularity among enterprises since it does not lead to vendor lock-in.

For the experimental investigation, a 12 node cluster has been chosen, designed and configured for large text format data processing. There two nodes are name nodes running in a high-available manner. This is an advisable number of master nodes recommended by Cloudera (Cloudera 2013). The remaining 10 data nodes run the worker roles for the Hadoop services. This is an empirically chosen number of data nodes.

Data nodes in the cluster have 4x Intel(R) Xeon(R) CPU E5–2680 v3 @ 2.50GHz, with 12x physical cores, 256 GB RAM, 10 TB HDD and Ethernet card each. Each node runs CentOS 6.7.

For the e several additional tools have been chosen: Hive version 1.10 (Hive-MR) on top of Hadoop 2.6.0-cdh5.4.8, Java version 1.6.0_31 and kite-dataset version

1.0.0-cdh5.4.8 to create a schema and dataset, import data from a text file, and view the results.

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.

B. Data used for experiments

Various databases and raw data examples exist. However, for the experiments a TPC-H (TPC 2014) database with a scale factor of 300 has been chosen due to its world-renowned characteristic. The scale factor of 300 means approximately 300 GB of data. An analysis shows that this is sufficient to provide insights into the advantages and limitations of each data format.

For data generation, a database population program DBGEN has been used. It is available on TPC website (TPC 2014) and designated for use with the TPC-H benchmark. As shown in Table 2, the TPC-H database consists of 8 separate and individual tables described in the TPC-H Benchmark Standard Specification Revision 2.17.1 (TPC 2014). All *.TBL files have been copied into HDFS as a plain text and converted to Avro and Parquet. For a shorter insight in the amount of data, the main table of TPC-H database (*lineitem.tbl*) consists of 1,799,989,091 rows and 16 columns. It is 230 MB large in plain text format (*.tbl), 116 MB large in Avro and 72 MB large in Parquet format.

A "put" command has been used to load data in to Hadoop distributed file system (HDFS). Fig. 1 shows an example of it for one of the tables in plain text format (*region.tbl*).

```
hdfs dfs –put region.tbl hdfs://tpc/data/
```

Fig. 1. Command line example used for data load into HDFS

After data load in to Hadoop, a *kite-dataset* command line (Apache 2015) has been used to convert data from the plain text format to Avro and Parquet format. The experiments have been performed with the default compression algorithm snappy for Avro and Parquet format because snappy compression provides a slightly better query performance than *zlib* and *gzip* (Floratou *et al.* 2014). Fig. 2 shows an example of kite-dataset commands used for plain text data converting to Avro and Parquet for one of the smallest TPC-H database table (*region.tbl*).

By default, *kite-dataset* supports converting from CSV and JSON formats. Thus a *csv-schema* argument has been used for data schema creation and a *csv-import* argument has been used for data import accordingly in Avro or Parquet format because original data has pipe delimited ("|") *.tbl format that is similar to delimiter separated values (DSV). Considering the fact that generated data files have lack of header, field names have been added with header argument in accordance with TPC-H data schema (TPC 2014).

Table 2. TPC-H table original size vs Avro and Parquet

TPC table name	Record count	*.tbl size MB	*.avro size MB	*.parquet size MB
customer.tbl	45,000,000	7,069.6777	3,971.8981	3,633.9168
lineitem.tbl	1,799,989,091	230,545.6467	116,639.3754	72,130.2250
nation.tbl	25	0.0021	0.0018	0.0028
orders.tbl	450,000,000	51,361.8456	24,943.3918	19,646.2062
partsupp.tbl	240,000,000	35,184.6488	14,446.4901	12,978.3418
part.tbl	60,000,000	7,040.0864	3,170.4650	1,843.1135
region.tbl	5	0.0004	0.0008	0.0014
supplier.tbl	3,000,000	410.8828	244.3105	231.1390
Total	–	331,612.7905	163,415.9335	110,462.9465

```
kite-dataset csv-schema hdfs://tpc/data/region.tbl –output hdfs://tpc/schemas/region.avsc --delimiter '|' --class TPC
--header 'regionkey|name|comment'
kite-dataset create dataset:hdfs://tpc/datasets/region_a -f avro --schema hdfs://tpc/schemas/region.avsc
kite-dataset csv-import hdfs://tpc/data/region.tbl dataset:hdfs://tpc/datasets/region_a --delimiter '|'
--header 'regionkey|name|comment'
kite-dataset create dataset:hdfs://tpc/datasets/region_p -f parquet --schema hdfs://tpc/schemas/region.avsc
kite-dataset csv-import hdfs://tpc/data/region.tbl dataset:hdfs://tpc/datasets/region_p --delimiter '|'
--header 'regionkey|name|comment'
```

Fig. 2. Example of command lines used for plain text data converting to Avro and Parquet

```
{"type" : "record",
"name" : "TPC",
"doc" : "Schema generated by Kite",
"fields" : [
{ "name" : "regionkey",
"type" : [ "null", "long" ],
"doc" : "Type inferred from '0'",
"default" : null
}, {
"name" : "name",
"type" : [ "null", "string" ],
"doc" : "Type inferred from 'AFRICA'",
"default" : null
}, {
"name" : "comment",
"type" : [ "null", "string" ],
"doc" : "Type inferred from 'lar depo'",
"default" : null }]}
```

Fig. 3. Data schema example of the smallest dataset (region.tbl)

The main table of TPC-H database (*lineitem.tbl*) consists of 1,799,989,091 rows and 16 columns. Although all *.TBL files have been copied into HDFS as a plain text for a shorter table schema insight the smallest table (*region.tbl*) has been chosen. Fig. 3 shows data schema for the smallest table (*region.tbl*).

The same schema (*.avsc) automatically created by kite-dataset *csv-schema* command has been chosen for data import into both formats (Avro and Parquet).

C. Data Load into Hive

Data was loaded into hive table by *CREATE TABLE* statement with "*stored as TEXTFILE*", "*stored as AVRO*" or "*stored as PARQUET*" accordingly to each dataset location. Fig. 4 shows *CREATE TABLE* statement syntax for the main table (*lineitems.tbl*) stored as Parquet.

The total count of tables created in Hive database is 24 accordingly to each of 8 TPC-H datasets and each of the three formats used for the experiments.

```
CREATE EXTERNAL TABLE dbase.tpc_lineitem_parq(
orderkey BIGINT, partkey BIGINT,
suppkey BIGINT, linenumber BIGINT,
quantity BIGINT, extendedprice DOUBLE,
discount DOUBLE, tax DOUBLE,
returnflag STRING, linestatus STRING,
shipdate STRING, commitdate STRING,
receiptdate STRING, shipinstruct STRING,
shipmode STRING, comment STRING)
STORED AS PARQUET
LOCATION 'hdfs://tpc/datasets/lineitem_p';
```

Fig. 4. CREATE TABLE statement example for *lineitem* data in Parquet format

D. Queries

The queries from TPC-H Benchmark (TPC 2014) have been mostly used for the experiments. Compiling statement and unsupported SubQuery Expression errors have been received during some TPC-H query execution. Thus, these queries have been rewritten to be useful for experiments. Modified queries are published in GitHub (DaigaPlase 2016) and are appropriately marked in Table 3. One of the modified queries (Q1) is showed in Fig. 5.

```
SELECT
RETURNFLAG, LINESTATUS,
SUM(QUANTITY) as sum_qty,
SUM(EXTENDEDPRICE) as sum_base_price,
SUM(EXTENDEDPRICE*(1-DISCOUNT)) as
sum_disc_price,
SUM(EXTENDEDPRICE*(1-DISCOUNT)*(1+TAX)) as
sum_charge,
AVG(QUANTITY) as avg_qty,
AVG(EXTENDEDPRICE) as avq_price,
AVG(DISCOUNT) as avg_discount,
COUNT(*) as count_order
FROM
dbase.tpc_lineitem_avro
WHERE
to_date(SHIPDATE)<='1996–07–02'
GROUP BY RETURNFLAG, LINESTATUS;
```

Fig. 5. Modified query 1 to select data from Avro formatted *lineitem* table based on TPC-H Q1

Basically, it is the same query that is described in TPC-H Benchmark. The modification is related with 'where' clause "l_shipdate <= date '1998–12–01' - interval '[DELTA]' day (3)" where the date interval has been replaced with the exact date and function to_date() in order to return the date from string type date value stored in Hive table, because data load into Hive without workaround approach of at least 4 steps (create temp table, load data, create table with correct data types and insert data there from temp table) supports only string type date values.

In addition, query 0 and query x23 have been added to TPC-H 22 query list for following purposes.

Query 0 has been defined simply for test purpose in order to check if the record count of each hive table corresponds to row count of each original *.tbl file. To count rows of each original data table command "*sed*" has been used, for example "*sed -n '$=' lineitem.tbl*" to output row count of *lineitem* table. Fig. 6 shows Query 0 used as aggregation query to examine Parquet advantage and count records from *lineitem* table of all three formats (stored as TEXTFILE, AVRO and PARQUET).

The output results that have been received with *sed* command and *count*(*) queries match. In addition, Query 0 execution time has been measured and included in Table 3 to illustrate performance of one simple aggregation function executed on different format tables.

```
select count(*) from dbase.tpc_lineitem_dsv
select count(*) from dbase.tpc_lineitem_avro
select count(*) from dbase.tpc_lineitem_parq
```

Fig. 6. Query 0 used as aggregation query to examine Parquet advantage and count records from lineitem table stored as TEXTFILE, AVRO and PARQUET

Query x23 has been defined as scan query for Avro format use case (Fig. 7), e. g., row-oriented data access, when only some columns are as interest of the processing. Query x23 does not include any aggregation.

```
select c.name, c.address from tpc_customer_dsv c where
c.acctbal=100;
select c.name, c.address from tpc_customer_avro c where
c.acctbal=100;
select c.name, c.address from tpc_customer_parq c where
c.acctbal=100;
```

Fig. 7. Query x23 used to examine Avro (SCAN) advantage

In the experiments, 22 TPC-H queries and these two additional queries have been executed, one after the other for plain text, Avro and Parquet formatted Hive table. The execution time has been measured for each query. Three full runs have been performed for each file format and each query. Thus, for each query, the average response time across the three runs has been reported.

Results

Data load in to Hadoop and conversion from the plain text format to Avro and Parquet format (Table 2) present significant storage space economy. Fig. 8 shows that the same data takes 2 times less storage space in Avro format, and 3 times less – in Parquet format. This is an answer to the second research question RQ.2.

The second research question related with null hypothesis H_0^C proves alternative hypothesis H_1^C that there is a difference in the compactness between data format Avro and Parquet, e. g., probability $\boldsymbol{p}$ that Parquet is more compact than Avro, $H_1^C : \boldsymbol{p}\left(A^C > P^C\right) \neq \boldsymbol{p}\left(P^C > A^C\right)$.

Although the data format Avro and Parquet use the same compression Snappy, the difference between Avro and Parquet shows that Parquet is approximately 1.5 times more compact than Avro.

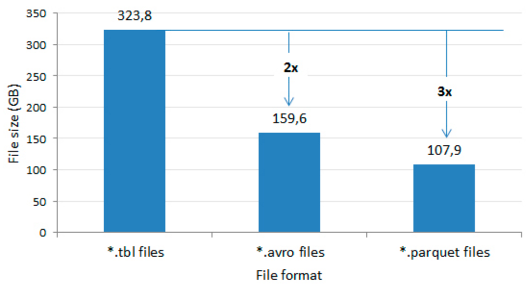

Fig. 8. Data size comparison between three formats

The answer to the first research question RQ.1 has been gained by performing the experiments and measuring execution time of 24 queries by using Beeline shell.

Beeline's reported time is close to time reported by Cloudera Resource manager for the same query. In addition, for data validation purposes shell script has been written in order to compare Beeline's reported time with shell output between two timestamps (query end time and start time). The shell time for each query is approximately 4 s higher than Beeline's time. This margin is because of the time required for query start and end.

In the experiments, 24 queries have been executed for each table (stored respectively as Textfile, Avro and Parquet). Table 3 presents the running time of the queries for each file format used for the experiments. In addition, Table 3 presents how many times the Parquet format is faster than Textfile and Avro respectively. Modified TPC-H queries are appropriately marked with (*) except Q0 and Qx23 that are new queries defined separately.

As shown in Table 3 and Fig. 9, Parquet can provide 2 times faster execution time on average when compared with Avro and Textfile.

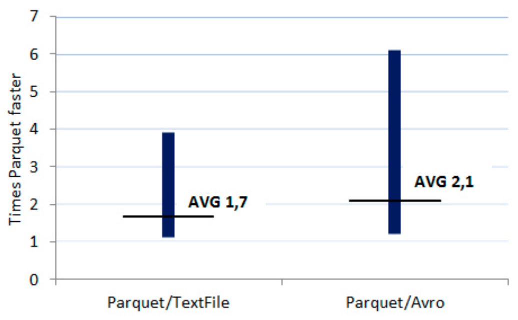

Fig. 9. Times Parquet faster Textfile and Avro (on average of all queries)

In order to answer the first research question, queries have been grouped into two parts accordingly to hypothesis H_0^A and H_0^B: 1) scan queries (Q2, Q3, Q4, Q20, Qx23); 2) the remaining (aggregation) queries.

Table 3. Query execution time (s, ms) and Parquet performance evaluation

TPC-H Query*	Aggregation (AGR) or SCAN query	Data format (Hive table 'stored as')			Times Parquet faster in comparison	
		Textfile (*.tbl)	Avro	Parquet	Textfile / Parquet	Avro / Parquet
Q0*	AGR	132,724	209,394	34,398	3,9	6,1
Q1*	AGR	306,444	321,427	142,364	2,2	2,3
Q2	SCAN	FAILED	FAILED	FAILED		
Q3*	SCAN	429,944	499,45	277,121	1,6	1,8
Q4*	SCAN	351,12	395,957	207,366	1,7	1,9
Q5*	AGR	506,531	557,565	324,148	1,6	1,7
Q6*	AGR	146,756	234,58	64,656	2,3	3,6
Q7*	AGR	633,338	664,841	436,435	1,5	1,5
Q8	AGR	FAILED	FAILED	FAILED		
Q9	AGR	FAILED	FAILED	FAILED		
Q10*	AGR	403,579	465,389	230,908	1,7	2,0
Q11	AGR	325,108	319,164	276,336	1,2	1,2
Q12*	AGR	325,803	359,147	182,783	1,8	2,0
Q13	AGR	216,121	244,936	201,872	1,1	1,2
Q14*	AGR	275,926	315,728	154,344	1,8	2,0
Q15*	AGR	608,079	675,436	325,472	1,9	2,1
Q16*	AGR	281,495	298,717	238,782	1,2	1,3
Q17	AGR	609,197	690,604	344,03	1,8	2,0
Q18	AGR	688,337	800,813	428,181	1,6	1,9
Q19	AGR	FAILED	FAILED	FAILED		
Q20*	SCAN	542,506	645,825	391,9	1,4	1,6
Q21*	AGR	1002,767	1266,491	678,115	1,5	1,9
Q22	AGR	215,96	295,432	152,604	1,4	1,9
Qx23*	SCAN	28,169	55,592	25	1,1	2,2
AVERAGE					1,7	2,1

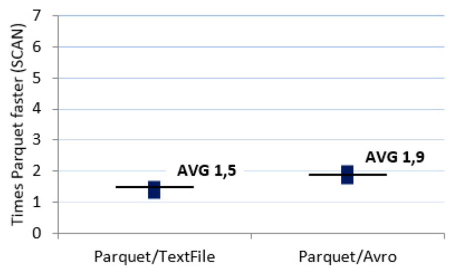

Fig. 10. Times Parquet faster Textfile and Avro (on average to SCAN queries)

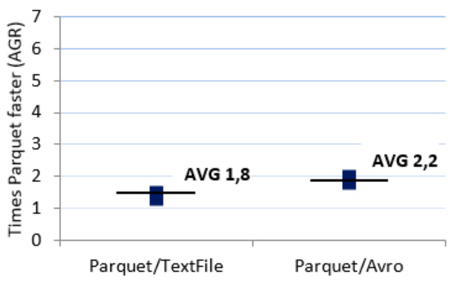

Fig. 11. Times Parquet faster Textfile and Avro (on average to AGGREGATION queries)

As shown in Fig. 10 and Fig. 11, Avro presents the worst performance when compared with Textfile and Parquet on both kind of queries (scan and aggregation). There is an insignificant difference between scan queries presented in Fig. 10 and aggregation queries presented in Fig. 11.

Thus, there is wrong null hypothesis H_0^A that data format Avro is better than Parquet in performance to scan queries because data format Parquet performs better than Avro on both kinds of queries, e. g. scan and aggregation queries. Thereby the null hypothesis H_0^B is true.

Summary and conclusions

1. The experiments performed within the scope of this article have been based on a systematic review of SQL-on-Hadoop by using compact data formats (Plase 2016). As the result of systematic literature review, a gap and need for additional experiments and studies have been formulated in order to answer the research questions about Parquet and Avro format. All 17 studies analyzed at the last stage of the systematic literature review

(Plase 2016) are not containing direct focus on comparing two binary data storage formats – Parquet and Avro because of both design specifics. Parquet as stated in the official documentation (Apache 2013) is a column-oriented data storage format. Thus, it should provide better performance on column-oriented queries, e. g., when only a specific set of those is selected. As a counterpart, Avro format is designed for row-oriented data access, e.g., when all columns are the interest of processing. Considering this, three hypothesis have been formulated in this article.

2. The experiments show that Avro usage is worth only from storage space economy point of view. Queries from Avro tables are slower when compared with queries even from Textfile format tables. However, all TPC-H queries from Parquet format tables provide a significant performance advantage over Textfile and Avro. Parquet can provide 2 times faster execution time on average when compared with Avro and Textfile. There is an insignificant difference between scan queries presented and aggregation queries.
3. A great deal of work has been done on the experiments with TPC-H datasets. TPC-H decision support benchmarks are widely used today in evaluating the performance of relational database systems. TPC-H datasets are usable in evaluating the performance of Big Data management systems because DBGEN allows to generate datasets with scale factor more than 1 TB. As future work might be mentioned query performance measuring by TPC-DS standard benchmark what is more appropriate to Big Data systems. In addition, other query engines like Impala, HAWQ, IBM Big SQL, Drill, Tajo, Pig, Presto and frameworks like Spark, Cascading, and Crunch could be considered for new experiments in order to gain more detailed experience with compact data formats.
4. The topic about data formats is related with work experience in Big Data field. There are many companies that manage Big Data (mostly and at this moment – banks, telecommunication, travel and tourism companies) and asking to define best practices for Avro and Parquet utilization. In addition, the result of previously done systematic review of SQL-on-Hadoop by using compact data formats (Plase 2016) and recognized research gap has been a motivation source for this research paper.

Acknowledgements

The authors would like to thank Accenture Latvia for providing the infrastructure used for the experiments.

References

Abadi, D. J.; Boncz, P. A.; Harizopoulos, S. 2009. Column-oriented database systems, *Processing of the VLDB Endowment* 2(2): 1664–1666. https://doi.org/10.14778/1687553.1687625

Apache. 2009a. *Avro specification* [online], [cited 30 November 2016]. Avro. Available from Internet: http://avro.apache.org/docs/current/spec.html

Apache. 2009b. *Sequence file* [online], [cited 30 November 2016]. Hadoop Hive. Available from Internet: https://wiki.apache.org/hadoop/SequenceFile

Apache. 2009c. *Thrift* [online], [cited 30 November 2016]. Apache Thrift. Available from Internet: http://thrift.apache.org

Apache. 2013. *Parquet official documentation* [online], [cited 30 November 2016]. Parquet. Available from Internet: https://parquet.apache.org/documentation/latest/

Apache. 2015. *Kite Dataset command line interface documentation* [online], [cited 30 November 2016]. Kite Software Development Kit. Available from Internet: http://kitesdk.org/docs/1.1.0/cli-reference.html

Apache. 2017. *Language manual ORC* [online], [cited 30 November 2016]. Apache Hive. Available from Internet: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC

Bray, T. 2014. *The JavaScript object notation (JSON) data interchange format* [online], [cited 30 November 2016]. Google, Inc. Available from Internet: https://tools.ietf.org/html/rfc7159

Cejka, S.; Mosshammer, R.; Einfalt, A. 2015. Java embedded storage for time series and meta data in Smart Grids, in *Proceedings of IEEE International Conference on Smart Grid Communications (SmartGridComm),* 2–5 November, 2015, Miami, USA, 434–439. https://doi.org/10.1109/smartgridcomm.2015.7436339

Chandra, D. G.; Prakash, R.; Lamdharia, S. 2012. A study on cloud database. Computational Intelligence and Communication Networks (CICN), in *Proceedings of Fourth International Conference on Computational Intelligence and Communication Networks, IEEE,* 3–5 November, 2012, Mathura, India, 513–519. https://doi.org/10.1109/cicn.2012.35

Chen, Y.; Qin, X.; Bian, H.; Chen, J.; Dong, Z.; Du, X.; Zhang, H. 2014. A study of SQL-on-Hadoop systems, in J. Zhan, R. Han, C. Weng (Eds.). *Big data benchmarks, performance optimization, and emerging hardware*. BPOE 2014. Lecture notes in Computer Science, Vol. 8807. Springer International Publishing, 154–166. https://doi.org/10.1007/978–3–319–13021–7_12

Cloudera. 2013. *How-to: select the right hardware for your new Hadoop cluster* [online], [cited 30 November 2016]. Available from Internet: https://blog.cloudera.com/blog/2013/08/how-to-select-the-right-hardware-for-your-new-hadoop-cluster/

Shafranovich, Y. 2005. *Common format and MIME type for comma-separated values (CSV) files* [online], [cited 30 November 2016]. SolidMatrix Technologies, Inc. Available from Internet: https://tools.ietf.org/html/rfc4180

DaigaPlase. 2016. *Personal repository 'DaigaPlase' in GitHub*, [online], [cited 30 November 2016]. Git Hub. Available from Internet: https://github.com/DaigaPlase/tpc_hive.git

Floratou, A.; Minhas, F. U.; Özcan, F. 2014. SQL-on-Hadoop: full circle back to shared-nothing database architectures, *Processing of the VLDB Endowment* 7(12): 1295–1306. https://doi.org/10.14778/2732977.2733002

Gartner. 2014. *Gartner says smartphone sales surpassed one billion units in 2014* [online], [cited 30 November 2016]. Gartner. Available from Internet: http://www.gartner.com/newsroom/id/2996817

Google. 2001. *Protocol buffers* [online], [cited 30 November 2016]. Google. Available from Internet: https://github.com/google/protobuf

Grover, A.; Gholap, J.; Janeja, V. P.; Yesha, Y.; Chintalapati, R.; Marwaha, H.; Modi, K. 2015. SQL-like big data environments: case study in clinical trial analytics, in *Proceedings of 2015 IEEE International Conference on Big Data (Big Data),* 29 October–01 November, 2015, Santa Clara, USA, 2680–2689.

He, Y.; Lee, R.; Huai, Y.; Shao, Z.; Jain, N.; Zhang, X.; Xu, Z. 2011. RCFile: a fast and space-efficient data placement structure in MapReduce-based warehouse systems, in *Proceedings of IEEE 27th International Conference on Data Engineering (ICDE)*, 11–16 April, 2011, Hannover, Germany, 1199–1208. https://doi.org/10.1109/icde.2011.5767933

Luckow, A.; Kennedy, K.; Manhardt, F.; Djerekarov, E.; Vorster, B.; Apon, A. 2015. Automotive big data: applications, workloads and infrastructures, in *Proceedings of 2015 IEEE International Conference on Big Data (Big Data),* 29 October–01 November, 2015, Santa Clara, USA, 1201–1210.

Palmer, N.; Miron, E.; Kemp, R.; Kielmann, T.; Bal, H. 2011. Towards collaborative editing of structured data on mobile devices, in *Proceedings of 12th IEEE International Conference on Mobile Data Management (MDM),* 6–9 June, 2011, Lulea, Sweden, 1: 194–199. https://doi.org/10.1109/mdm.2011.48

Plase, D. 2016. *A systematic review of SQL-on-Hadoop by using compact data formats* [online], [cited 30 November 2016]. Preprint (MII). Available from Internet: https://dspace.lu.lv/dspace/handle/7/34452

Sharma, M.; Hasteer, N.; Tuli, A.; Bansal, A. 2014. Investigating the inclinations of research and practices in Hadoop: a systematic review. Confluence the next generation information technology summit (confluence), in *Proceedings of 5th International Conference – Confluence The Next Generation Information Technology Summit (Confluence 2014),* 25–26 September, 2014, Noida, India, 227–231. https://doi.org/10.1109/confluence.2014.6949381

Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. 2010. The hadoop distributed file system, in *Proceedings of IEEE 26th Symposium on Mass Storage Systems and Technologies* (MSST), 3–7 May, 2010, Lake Tahoe, USA, 1–10. https://doi.org/10.1109/msst.2010.5496972

Stonebraker, M.; Abadi, D. J.; Batkin, A.; Chen, X.; Cherniack M.; Ferreira M.; O'Neil, P. 2005. C-store: a column-oriented DBMS, in *Proceedings of the 31st international conference on Very large databases*, VLDB Endowment, August 30–September 2, 2005, Trondheim, Norway, 553–564.

Tapiador, D.; O'Mullane, W.; Brown, A. G. A.; Luri, X.; Huedo, E.; Osuna, P. 2014. A framework for building hypercubes using MapReduce, *Computer Physics Communications* 185(5): 1429–1438. https://doi.org/10.1016/j.cpc.2014.02.010

TPC. 2014. *TPC-H benchmark standard specification revision 2.17.1* [online], [cited 30 November 2016]. TPC. Available from Internet: http://www.tpc.org/tpc_documents_current_versions/current_specifications.asp

Wonjin, L.; On, B. W.; Lee, I.; Choi, J. 2014. A big data management system for energy consumption prediction models, in *Proceedings of 9th International Conference on Digital Information Management* (ICDIM), 29 September–01 October, 2014, Bankok, Thailand, 156–161.

Zhang, S.; Miao, L.; Zhang, D.; Wang, Y. 2014. A strategy to deal with mass small files in HDFS, in *Proceedings of 2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics* (IHMSC), 26–27 August, 2014, Hangzhou, Zhejiang, China, 1: 331–334. https://doi.org/10.1109/ihmsc.2014.87

HDFS **GLAUSTŲJŲ DUOMENŲ FORMATŲ PALYGINIMAS:** ***AVRO*** **PRIEŠ** ***PARQUET***

D. Plase, L. Niedrite, R. Taranovs

Santrauka

Straipsnyje vertinamas duomenų užklausų našumas lyginant *Avro* ir *Parguet* failų formatus su teksto failų formatu. Tyrimuose taikytos įvairios duomenų užklausų formos, naudota *Cloudera* atvirojo kodo *Apache Hadoop* CDH 5.4 versijos programinė įranga. Tyrimo rezultatai patvirtina, kad glaustieji duomenų formatai (*Avro* ir *Parguet*) dėl galimybės įterpti dvejetainį kodą ir naudoti glaudą taupo atmintį. Parodoma, kad duomenų užklausos įvykdomos sparčiau naudojant *Parquet* nei *Avro* ar teksto failų formatus.

Reikšminiai žodžiai: didieji duomenys, *Hadoop*, HDFS, *Hive*, *Avro*, *Parquet*.