LATVIJAS UNIVERSITĀTE
ANNO 1919

Accelerating Data Queries on Hadoop Framework by Using Compact Data Formats

Daiga Plase

Supervisor: prof., Dr. sc. comp.
Laila Niedrite

Presentation contents

1) Research problem, motivation, tasks
2) HDFS data formats
3) Tests and experiments
4) Conclusions
5) Future work

Problem (why Hadoop?)

- the traditional computing techniques – insufficient to process distributed and real-time large data sets- so-called Big Data
- huge volumes of raw data
  stored in specific text formats, for instance: JSON, CSV, XML, etc.
- a disk space need to store such files
- data storage format may impact the speed of data processing with Hadoop tools, like Hive

Motivation to sort out the problem

Where and how to store
these data in order
to provide database
for faster execution
of data queries?

Research task

to define how to utilize
Avro or Parquet
and find the best practices

Why Avro and Parquet?

File format	Schema integration	Compression support
Text/CSV	-	-
JSON	+	-
Avro	+	+
SequenceFile	-	+
RCFile	-	+
ORC file	-	+
Parquet	+	+

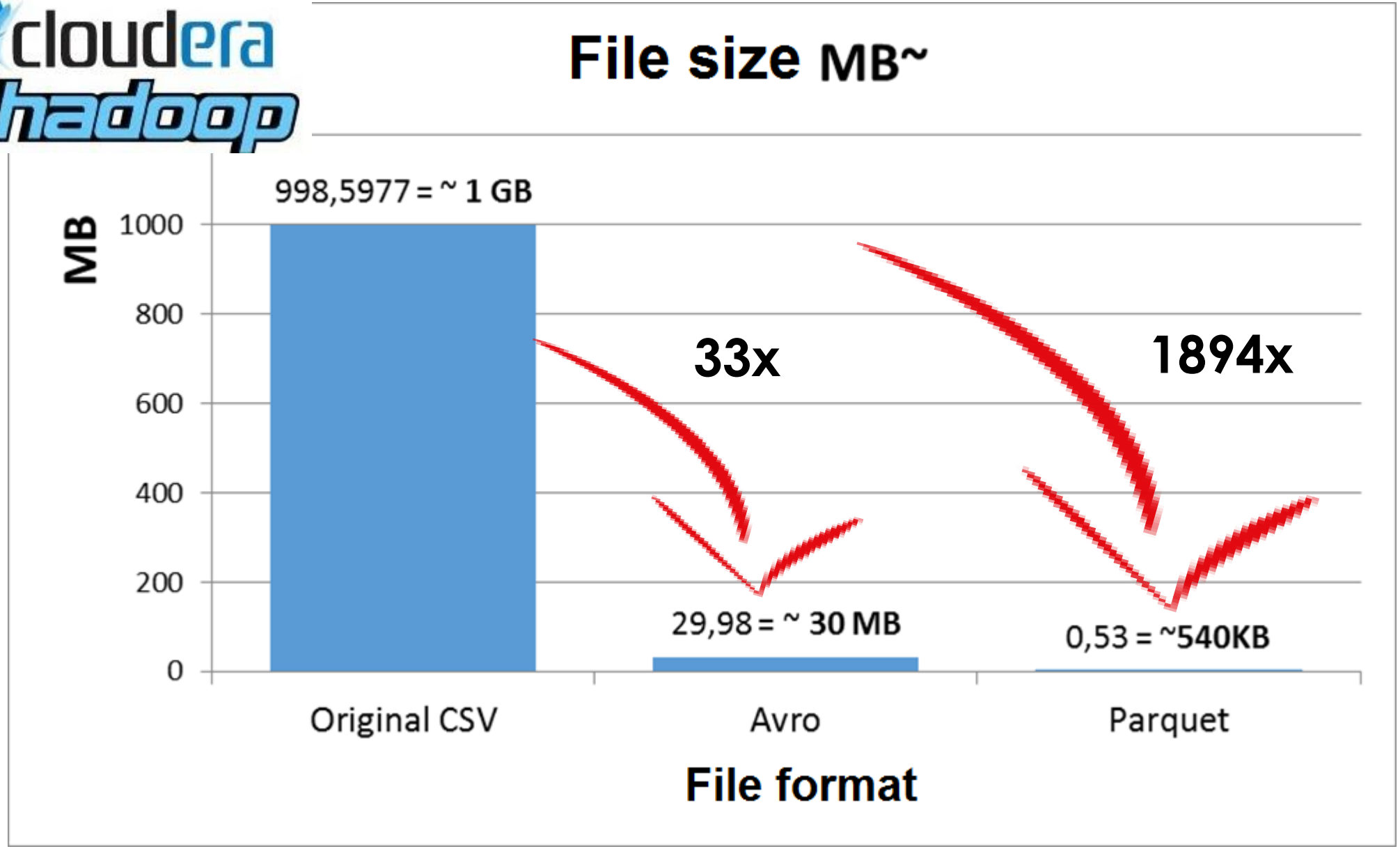
Research questions and method

- **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?

An Experiment

The first pre-test (Avro / Parquet in size)

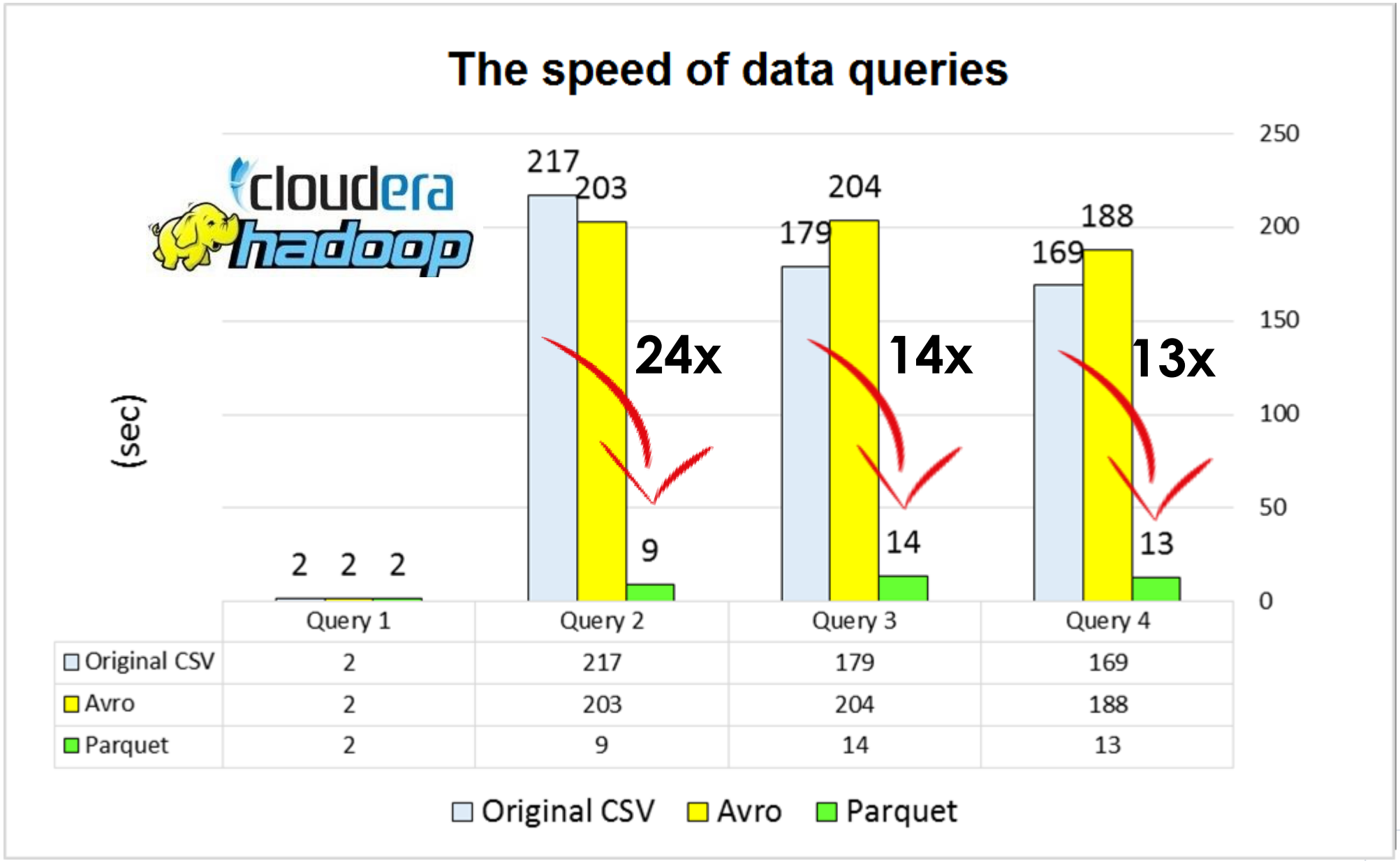


Data structure and queries

1) select * from dataset
2) select **count**(*) from dataset
3) select route, **SUM**(price_value) as pv from dataset **group** by route
4) select origin, **sum**(price_value) as pv from dataset where price_value< 100 **group** by origin

- **route STRING**,
- **origin** STRING,
- destination STRING,
- direct_flight STRING,
- type STRING,
- departure_flight STRING,
- arrival_time STRING,
- airlines STRING,
- duration STRING,
- price_type STRING,
- **price_value DOUBLE**,
- discount STRING,
- departure_date STRING,
- days_long_trip INT,
- date_when_gathered STRING,
- time_when_gathered STRING,
- flight_id STRING,
- seats_left STRING

Pre-Test result – the speed of data queries



Cluster configuration

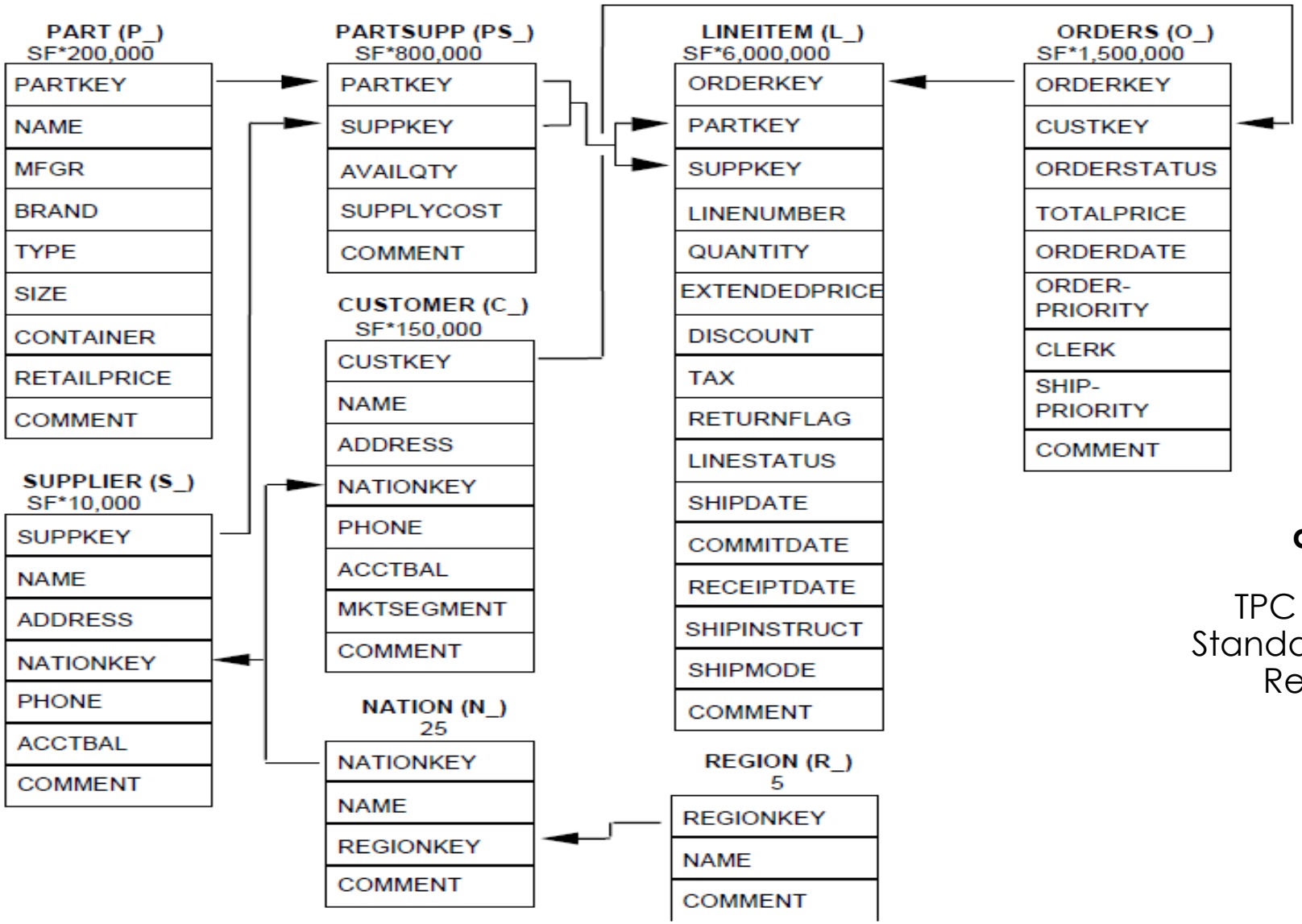
- **12** node cluster:
  - **2 name nodes** running in high-available manner
  - **10 data nodes** run the worker roles for the Hadoop services

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
- Ethernet card each

Each node runs CentOS 6.7.

Tools:
- Hive version 1.10 (Hive-MR) on top of Hadoop 2.6.0-cdh5.4.8
- Java version 1.6.0_31
- 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

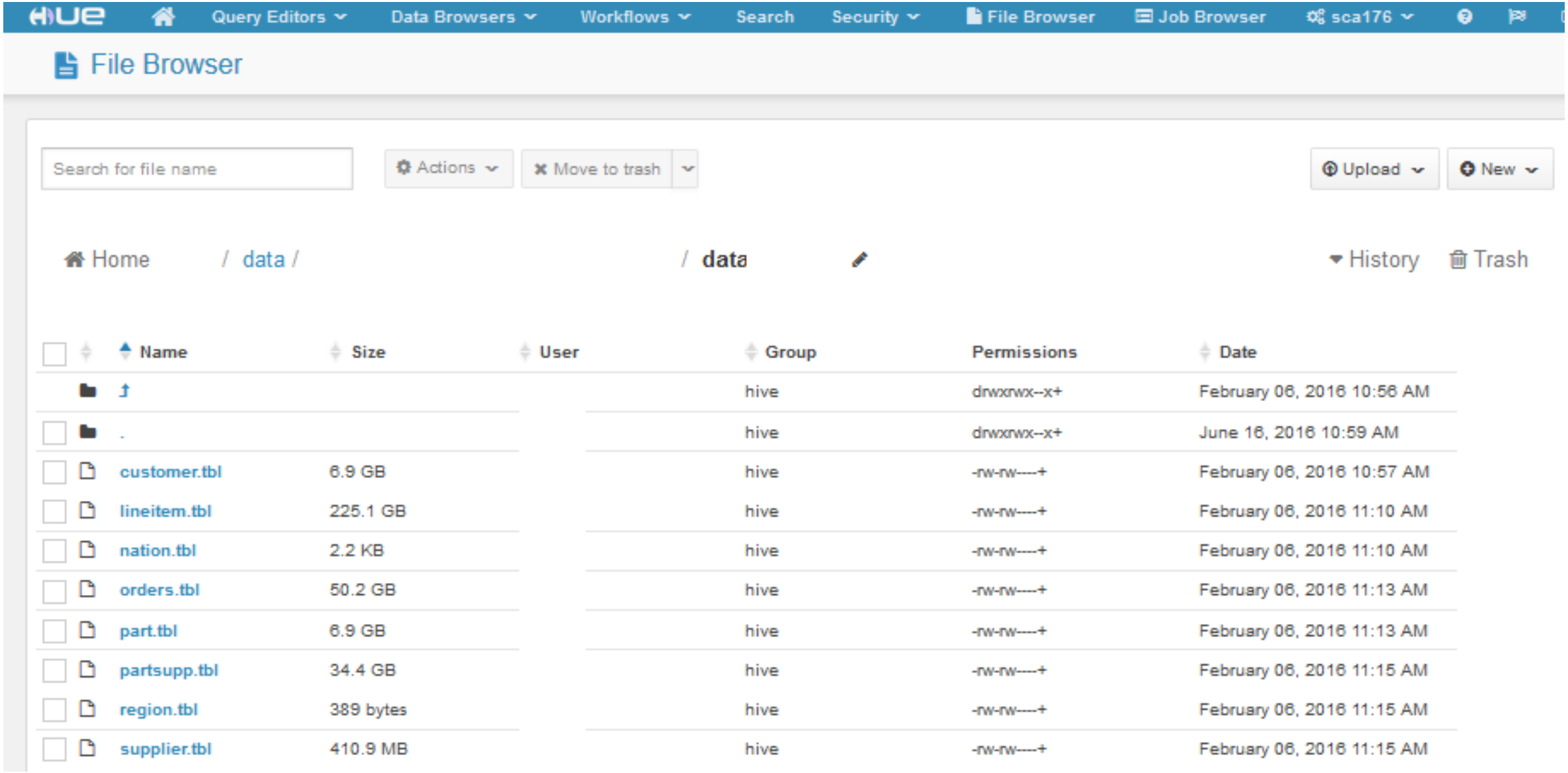
Data used for experiments



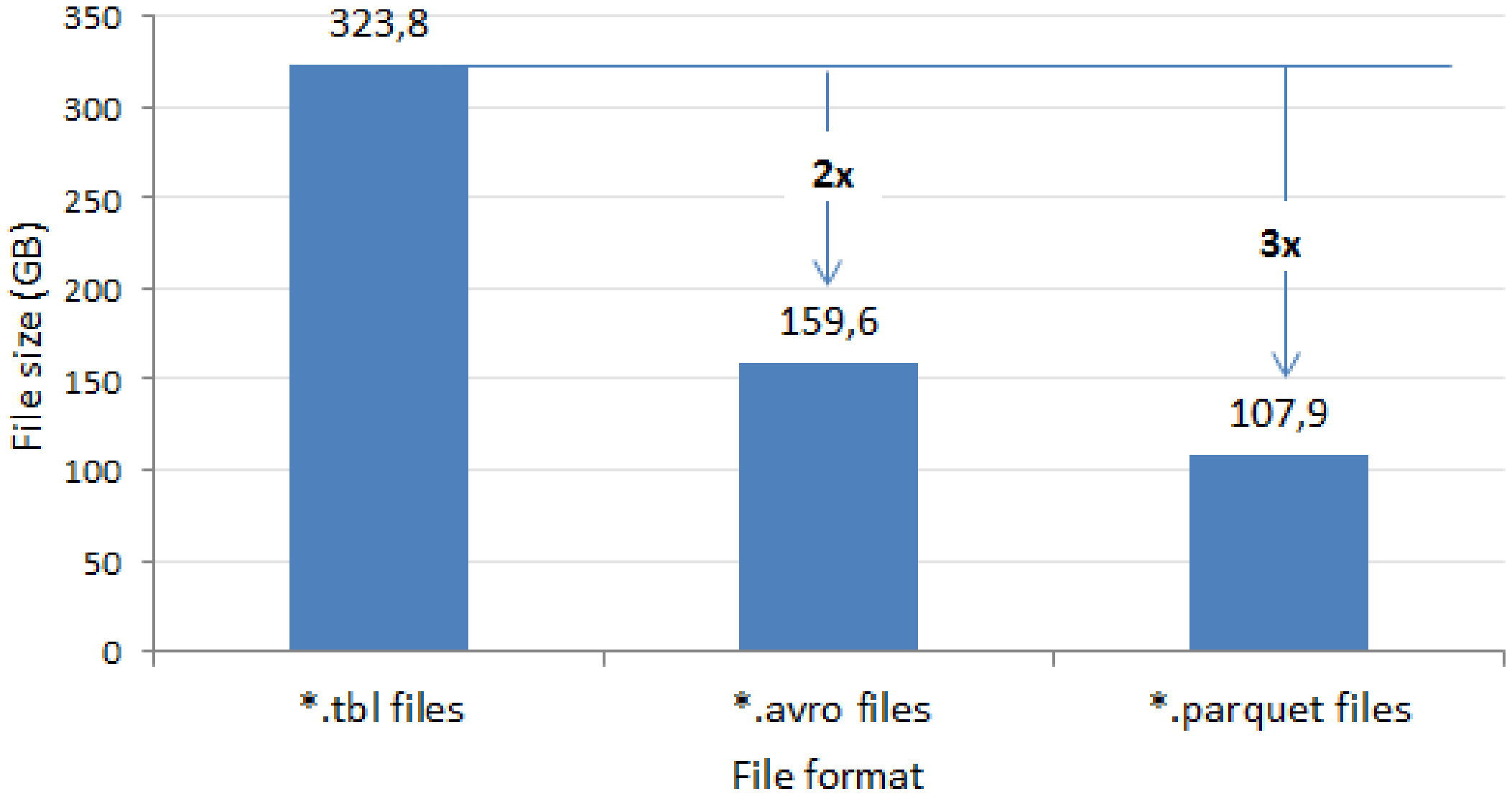
dbgen -s 300

TPC Benchmark H
Standard Specification
Revision 2.17.1

Data load into HDFS (Hue)



The result on compactness



Data queries & environment

```
-sh-4.1$ cd ~/test/queries/a6/
-sh-4.1$ ls
get_time1.sh      q11_p.sql  q13_t.sql  q16_a.sql         q18_a.sql        q1_t.sql          q21_t.sql         q3_a_result.txt  q4_t.sql          q6_p.sql          q9_p.sql          wq8_t.sql
q10_a_result.txt  q11_t.sql  q14_a.sql  q16_p.sql         q18_p.sql        q20_a.sql         q22_a.sql         q3_a.sql         q5_a_result.txt   q6_t.sql          time_result1.txt  wq9_a.sql
q10_a.sql         q12_a.sql  q14_p.sql  q16_t_result.txt  q18_t.sql        q20_p.sql         q22_p.sql         q3_p.sql         q5_a.sql          q7_a_result.txt   wq19_a.sql        wq9_t.sql
q10_p.sql         q12_p.sql  q14_t.sql  q16_t.sql         q19_p.sql        q20_t_result.txt  q22_t_result.txt  q3_t.sql         q5_p.sql          q7_a.sql          wq19_t.sql
q10_t.sql         q12_t.sql  q15_a.sql  q17_a.sql         q1_a_result.txt  q20_t.sql         q22_t.sql         q4_a_result.txt  q5_t.sql          q7_p.sql          wq2_a.sql
q11_a_result.txt  q13_a.sql  q15_p.sql  q17_p.sql         q1_a.sql         q21_a.sql         q2_p_result.txt   q4_a.sql         q6_a_result.txt   q7_t.sql          wq2_t.sql
q11_a.sql         q13_p.sql  q15_t.sql  q17_t.sql         q1_p.sql         q21_p.sql         q2_p.sql          q4_p.sql         q6_a.sql          q8_p.sql          wq8_a.sql
-sh-4.1$ cat get_time1.sh
cd /home/[redacted]test/queries/a6
source /home/[redacted]env.sh
echo "$(date) - ===START Q16_text==="
beeline -u $HIVE_URL -f q16_t.sql > q16_t_result.txt 2>&1
echo "$(date) - ===END Q16_text==="
```

```
-sh-4.1$ cat q1_a_result.txt
scan complete in 3ms
Connecting to jdbc:hive2://[redacted]:10000/;principal=hive/[redacted]
Connected to: Apache Hive (version 1.1.0-cdh5.4.8)
Driver: Hive JDBC (version 1.1.0-cdh5.4.8)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://[redacted]:10000/> USE dev_grp_base;
No rows affected (2.24 seconds)
0: jdbc:hive2://[redacted]:10000/> select
0: jdbc:hive2://[redacted]:10000/> l.returnflag,
0: jdbc:hive2://[redacted]:10000/> l.linestatus,
0: jdbc:hive2://[redacted]:10000/> sum(l.quantity) as sum_qty,
0: jdbc:hive2://[redacted]:10000/> sum(l.extendedprice) as sum_base_price,
um_disc_price,//[redacted]:10000/> sum(l.extendedprice * (1 - l.discount)) as s
+ l.tax)) as sum_charge,.67:10000/> sum(l.extendedprice * (1 - l.discount) * (1
0: jdbc:hive2://[redacted]:10000/> avg(l.quantity) as avg_qty,
0: jdbc:hive2://[redacted]:10000/> avg(l.extendedprice) as avg_price,
0: jdbc:hive2://[redacted]:10000/> avg(l.discount) as avg_disc,
0: jdbc:hive2://[redacted]:10000/> count(*) as count_order
0: jdbc:hive2://[redacted]:10000/> from
0: jdbc:hive2://[redacted]:10000/> tpc_lineitem_avro l
0: jdbc:hive2://[redacted]:10000/> where
0: jdbc:hive2://[redacted]:10000/> to_date(l.shipdate) <= '1998-09-16'
0: jdbc:hive2://[redacted]:10000/> group by
0: jdbc:hive2://[redacted]:10000/> l.returnflag,
0: jdbc:hive2://[redacted]:10000/> l.linestatus
0: jdbc:hive2://[redacted]:10000/> order by
0: jdbc:hive2://[redacted]:10000/> l.returnflag,
0: jdbc:hive2://[redacted]:10000/> l.linestatus;
INFO  : Number of reduce tasks not specified. Estimated from input data size: 1099
INFO  : In order to change the average load for a reducer (in bytes):
```

- **24** data queries in total
  - 22 TPC-H
  - 2 additional

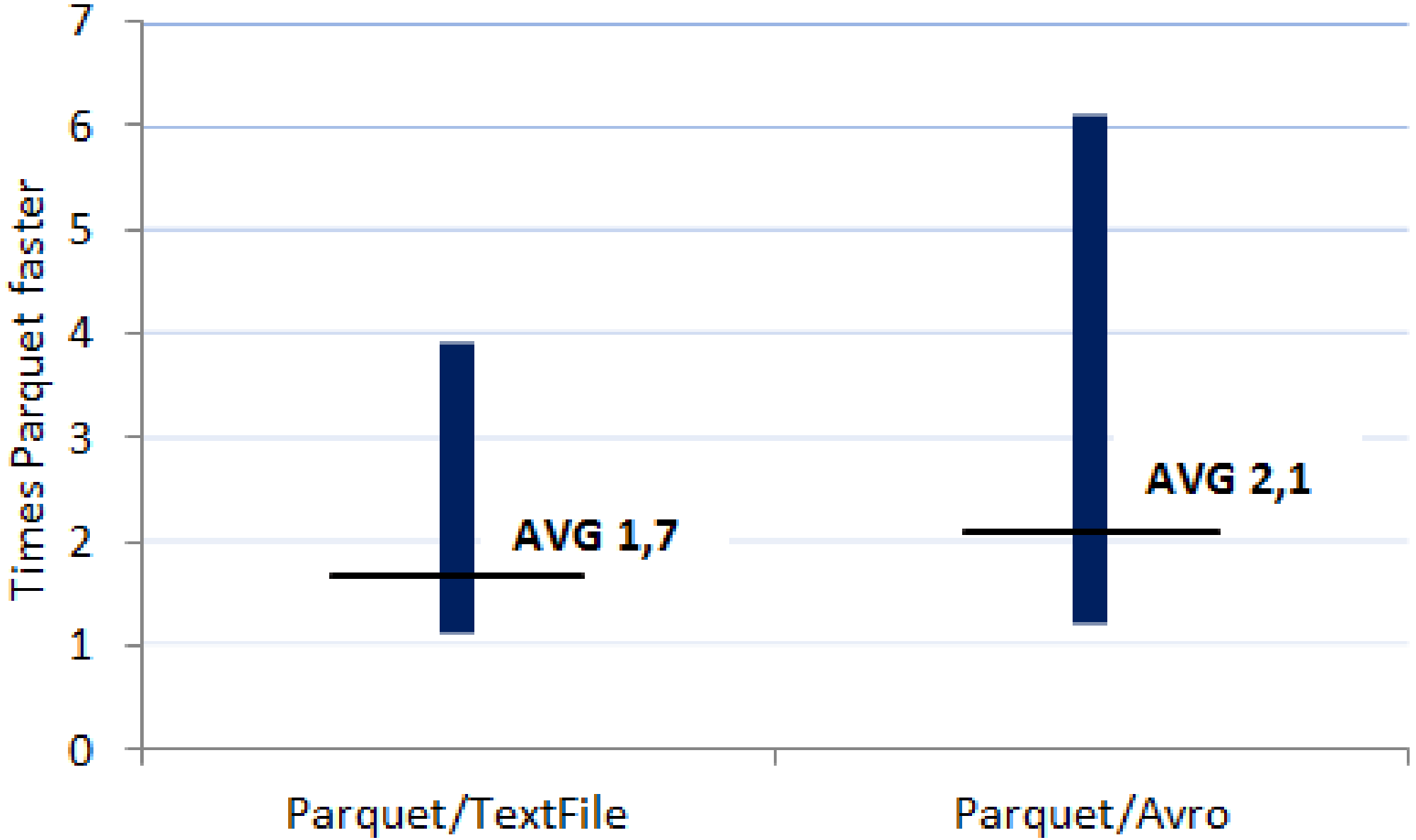
OR

- 5 AGGREGATION
- 19 SCAN queries

Summary table

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
Q11	AGR	325,108	319,164	276,336	1,2	1,2
Q12*	AGR	325,803	359,147	182,783	1,8	2
Q13	AGR	216,121	244,936	201,872	1,1	1,2
Q14*	AGR	275,926	315,728	154,344	1,8	2
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
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**

The result on Query execution time



Conclusions

- Data format **Parquet** performs **better** than Avro on **both kind of queries** (**scan** & **aggr**);
- There is **wrong** assumption **that** data format **Avro** is **better** than **Parquet** in performance on **scan** queries;
  Queries from Avro tables are slower when compared with queries even from Textfile format tables;
- **Avro** usage **is worth only from storage space economy** point of view;
  Parquet is 50% better in this point also.

Column vs row oriented storage format

Logical table example:

A	B	C
A1	B1	C1
A2	B2	C2
A3	B3	C3

Row-oriented layout (each row follows next)

A1	B1	C1	A2	B2	C2	A3	B3	C3

Instead **column-oriented** layout (one column is stored following next):

A1	A2	A3	B1	B2	B3	C1	C2	C3

Publications

✓ Accelerating data queries on Hadoop framework by using compact data formats

Advances in Information, Electronic and Electrical Engineering (AIEEE), 2016 IEEE 4th Workshop, 2016-11-12

Indexed Web of Science un SCOPUS: ISBN: 978-150904473-3
Source Type: Conference Proceeding
DOI: 10.1109/AIEEE.2016.7821807
Document Type: Conference Paper
Volume Editors: Romanovs A.,Navakauskas D.

Google Scholar:
http://ieeexplore.ieee.org/abstract/document/7821807/
Minēts: 3

✓ A Systematic Review of SQL-on-Hadoop by Using Compact Data Formats

Baltic Journal of Modern Computing, 2017-06-28
Indexed Web of Science
Link to article: http://www.bjmc.lu.lv/fileadmin/user_upload/lu_portal/projekti/bjmc/Contents/5_2_06_Plase.pdf
http://www.bjmc.lu.lv/contents/vol-52017-no-2/

✓ A Comparison of HDFS Compact Data Formats: Avro Versus Parquet

Science – Future of Lithuania / Mokslas – Lietuvos Ateitis, 2017-07-04
http://www.mla.vgtu.lt/index.php/mla/issue/view/54
ISSN 2029-2341, eISSN 2029-2252
DOI: 10.3846/mla.2017.1033
Link to article : http://www.mla.vgtu.lt/index.php/mla/article/download/1033/pdf

Future work

1) Deeper research of column-oriented data formats and data compression algotihms;
2) Research on Hadoop components and efficient combinations to increase data processing;
3) Tests and new experments;
4) To prepare the next scientific article;
5) Participation in scientific conference (AIEEE, DB&IS, others ...).

Questions?

E-mail: Daiga.Plase@gmail.com