File Format Benchmark - Avro, JSON, ORC, & Parquet

Owen O'Malley
owen@hortonworks.com
@owen_omalley

September 2016

Who Am I?

- Worked on Hadoop since Jan 2006
- MapReduce, Security, Hive, and ORC
- Worked on different file formats
  - Sequence File, RCFile, ORC File, T-File, and Avro requirements

HORTONWORKS

Goal

- Seeking to discover unknowns
  - How do the different formats perform?
  - What could they do better?
  - Best part of open source is looking inside!
- Use real & diverse data sets
  - Over-reliance on similar datasets leads to weakness
- Open & reviewed benchmarks

HORTONWORKS

The File Formats

HORTONWORKS

Avro

- Cross-language file format for Hadoop
- Schema evolution was primary goal
- Schema segregated from data
  - Unlike Protobuf and Thrift
- Row major format

HORTONWORKS

JSON

- Serialization format for HTTP & Javascript
- Text-format with MANY parsers
- Schema completely integrated with data
- Row major format
- Compression applied on top

HORTONWORKS

ORC

- Originally part of Hive to replace RCFile
  - Now top-level project
- Schema segregated into footer
- Column major format with stripes
- Rich type model, stored top-down
- Integrated compression, indexes, & stats

HORTONWORKS

Parquet

- Design based on Google's Dremel paper
- Schema segregated into footer
- Column major format with stripes
- Simpler type-model with logical types
- All data pushed to leaves of the tree
- Integrated compression and indexes

HORTONWORKS

Data Sets

HORTONWORKS

NYC Taxi Data

- Every taxi cab ride in NYC from 2009
  - Publically available
  - http://tinyurl.com/nyc-taxi-analysis
- 18 columns with no null values
  - Doubles, integers, decimals, & strings
- 2 months of data – 22.7 million rows

HORTONWORKS

HORTONWORKS

Github Logs

- All actions on Github public repositories
  - Publically available
  - https://www.githubarchive.org/
- 704 columns with a lot of structure & nulls
  - Pretty much the kitchen sink
- 1/2 month of data – 10.5 million rows

HORTONWORKS

Finding the Github Schema

- The data is all in JSON.
- No schema for the data is published.
- We wrote a JSON schema discoverer.
  - Scans the document and figures out the type
- Available at https://github.com/hortonworks/hive-json
- Schema is huge (12k)

HORTONWORKS

Sales

- Generated data
  - Real schema from a production Hive deployment
  - Random data based on the data statistics
- 55 columns with lots of nulls
  - A little structure
  - Timestamps, strings, longs, booleans, list, & struct
- 25 million rows

HORTONWORKS

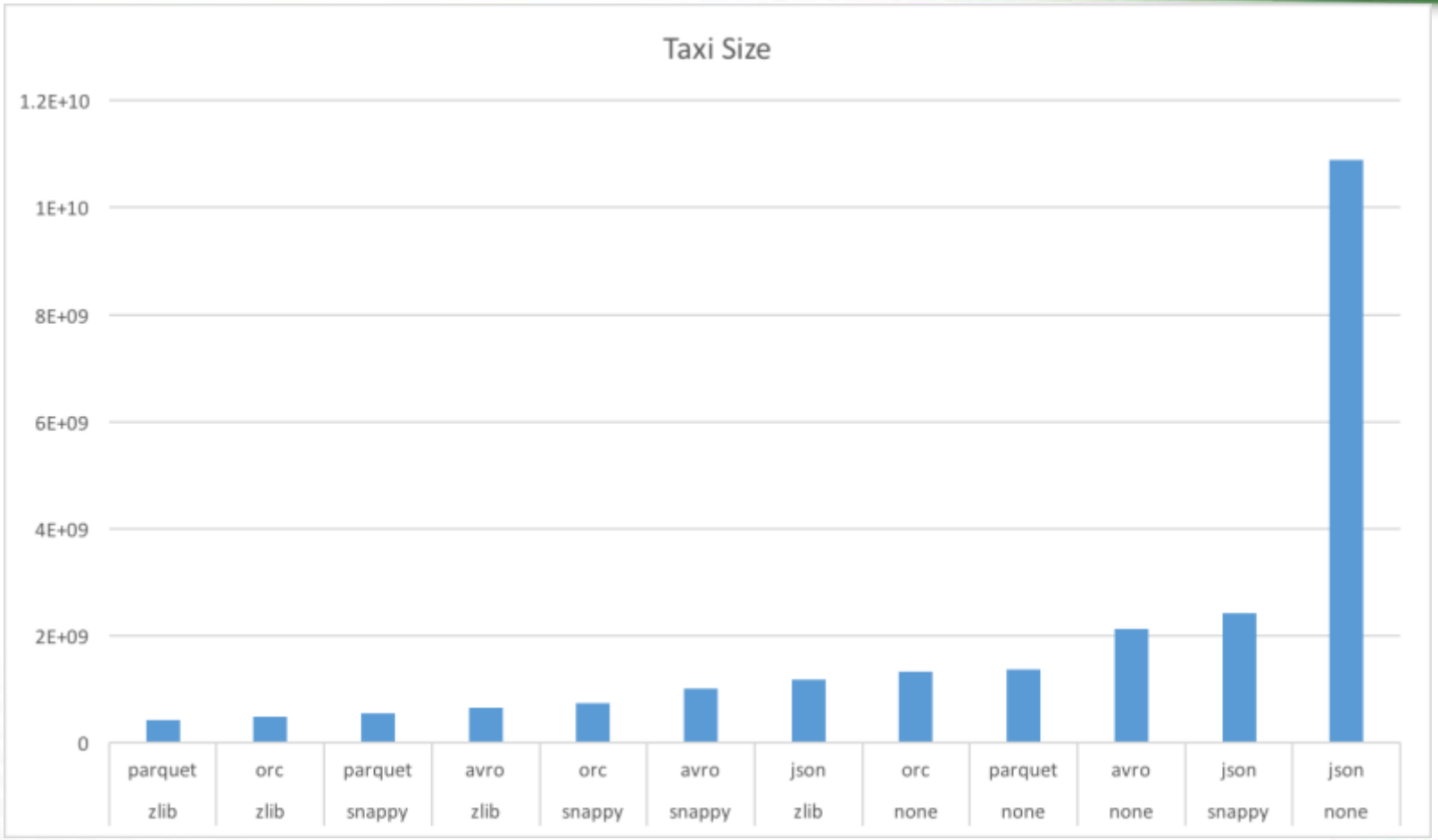
Storage costs

HORTONWORKS

Compression

- Data size matters!
  - Hadoop stores all your data, but requires hardware
  - Is one factor in read speed
- ORC and Parquet use RLE & Dictionaries
- All the formats have general compression
  - ZLIB (GZip) – tight compression, slower
  - Snappy – some compression, faster

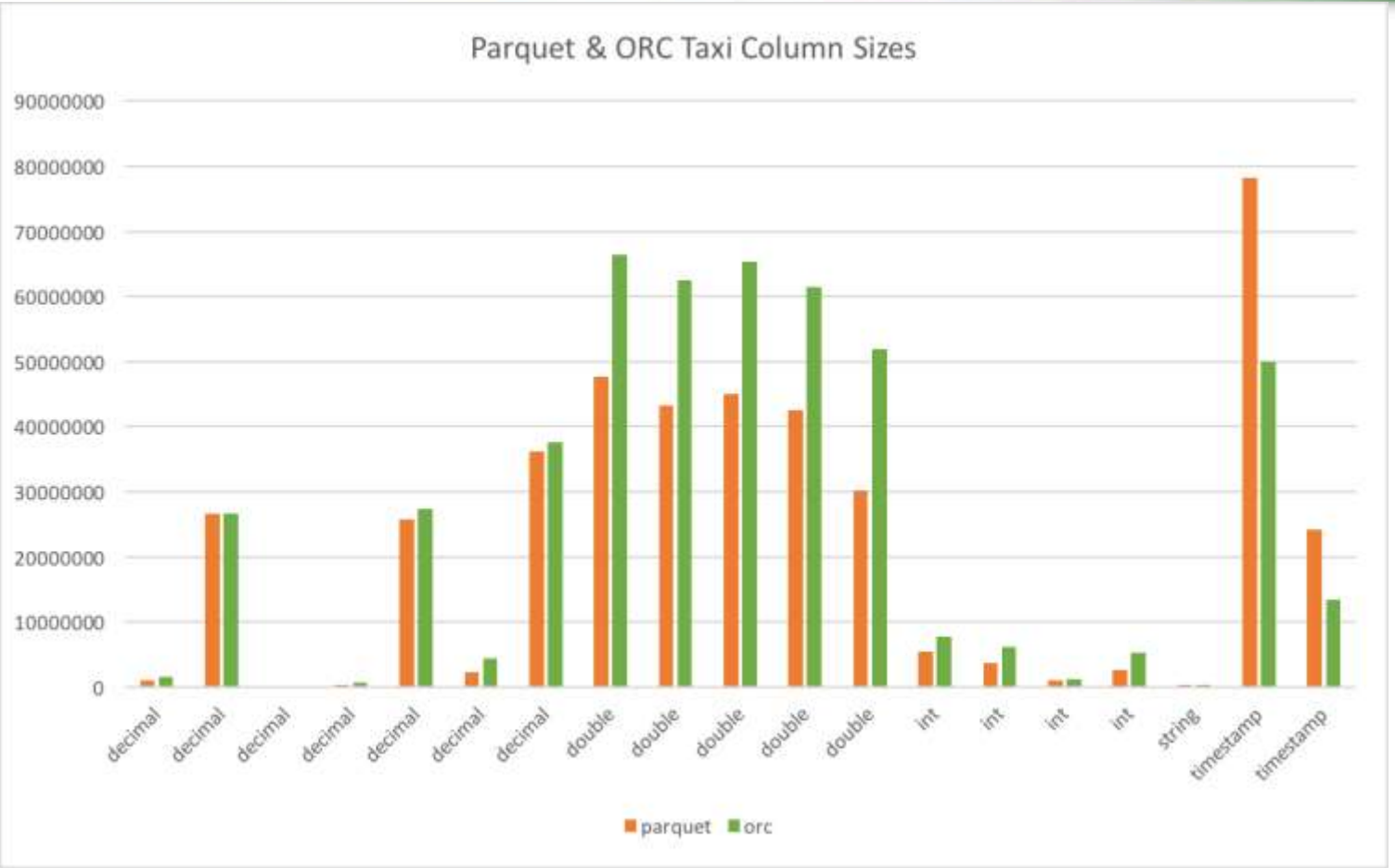
HORTONWORKS

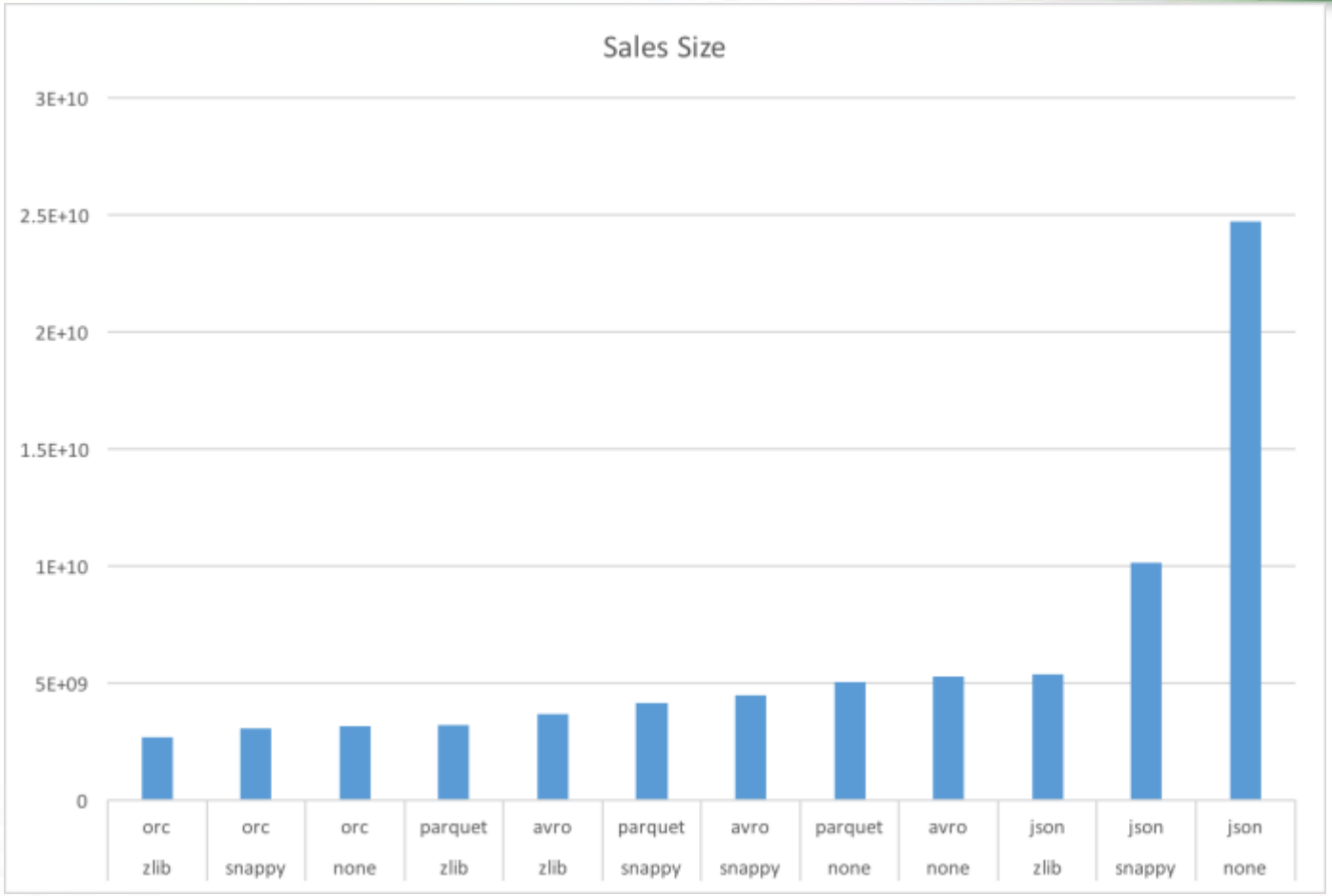


Taxi Size Analysis

- Don’t use JSON
- Use either Snappy or Zlib compression
- Avro’s small compression window hurts
- Parquet Zlib is smaller than ORC
  - Group the column sizes by type

HORTONWORKS

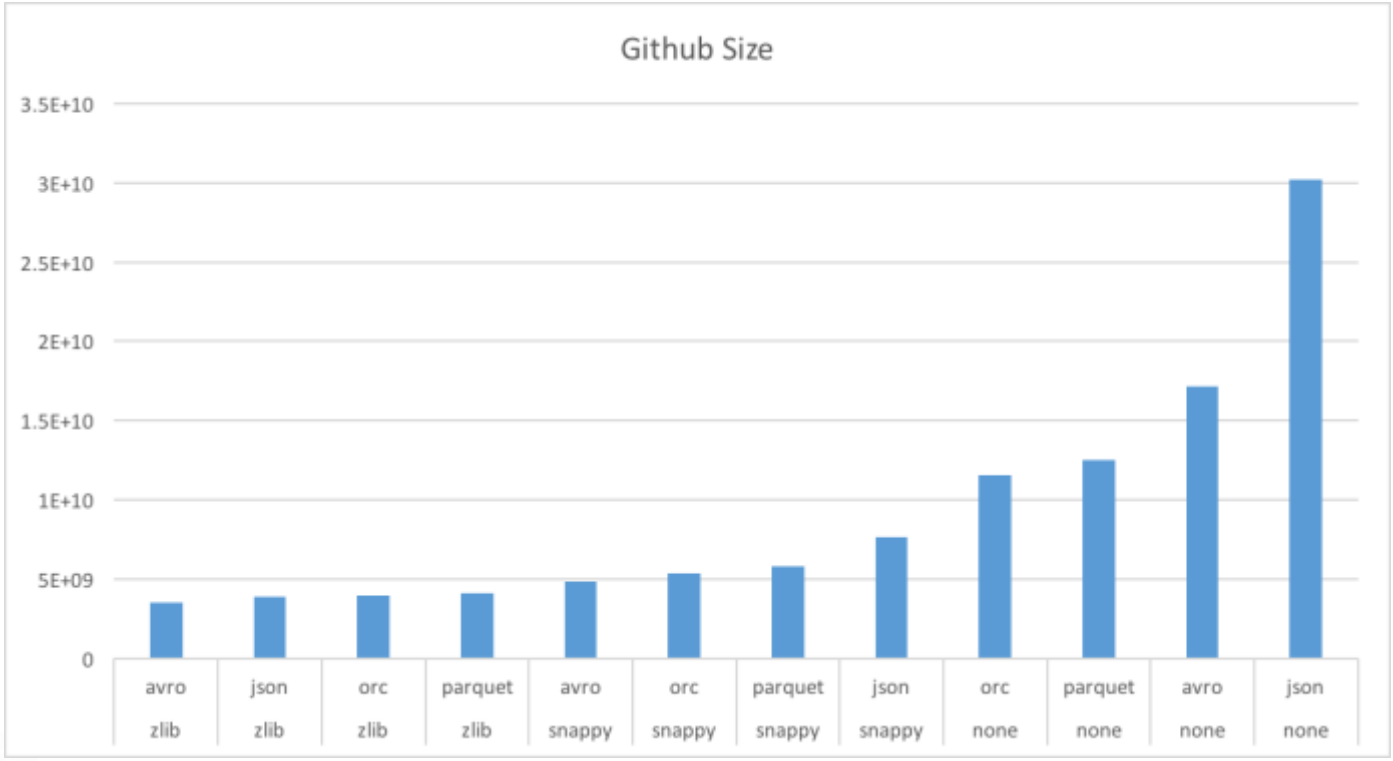




Taxi Size Analysis

- ORC did better than expected
  - String columns have small cardinality
  - Lots of timestamp columns
  - No doubles ☺
- Need to revalidate results with original
  - Improve random data generator
  - Add non-smooth distributions

HORTONWORKS



Github Size Analysis

- Surprising win for JSON and Avro
  - Worst when uncompressed
  - Best with zlib
- Many partially shared strings
  - ORC and Parquet don't compress across columns
- Need to investigate shared dictionaries.

HORTONWORKS

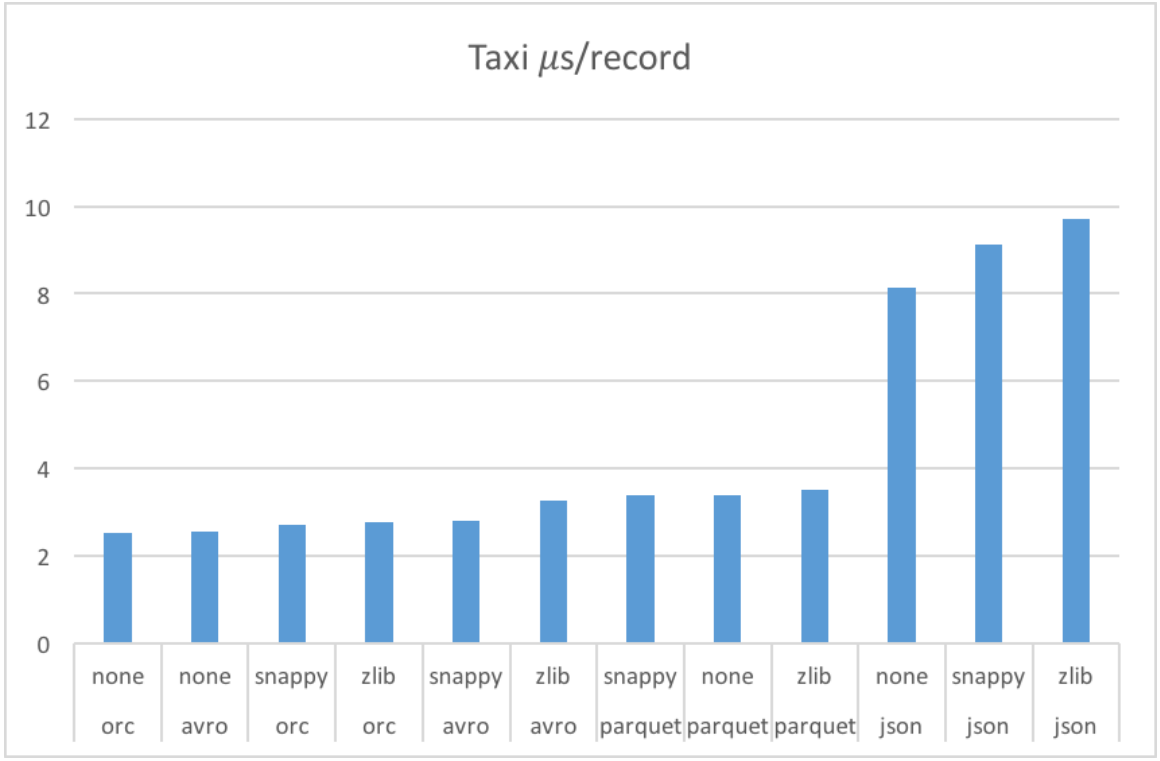
Use Cases

HORTONWORKS

Full Table Scans

- Read all columns & rows
- All formats except JSON are splitable
  - Different workers do different parts of file

HORTONWORKS

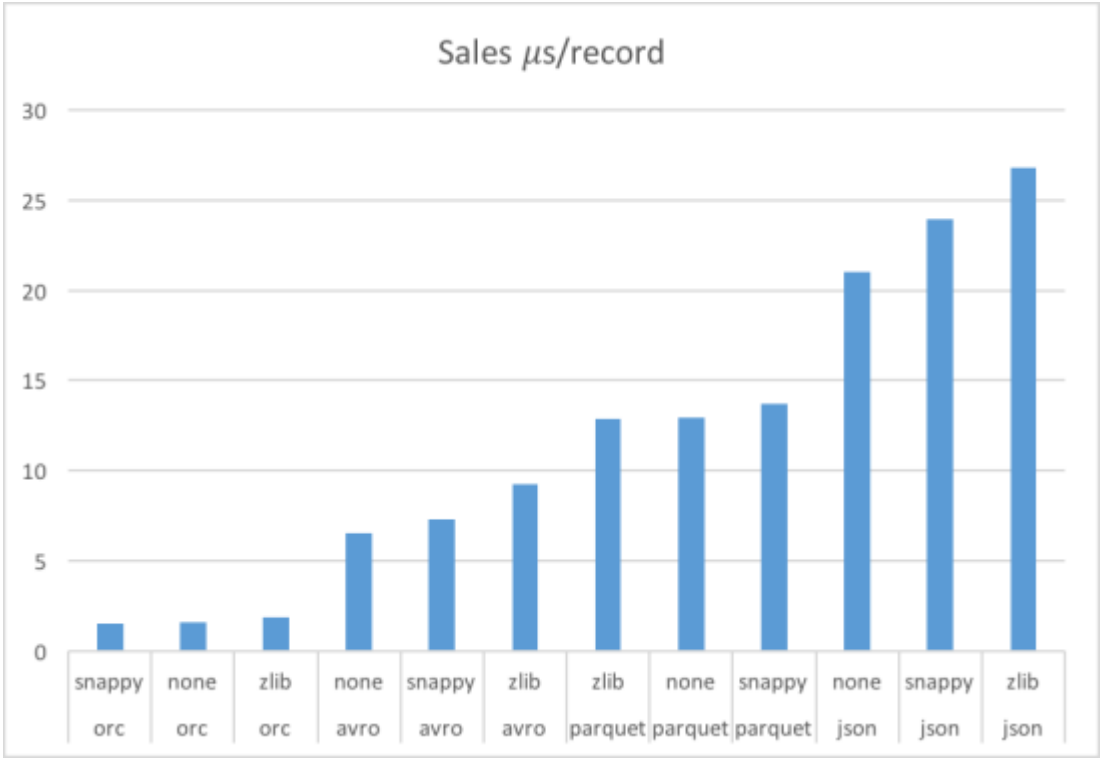


HORTONWORKS

Taxi Read Performance Analysis

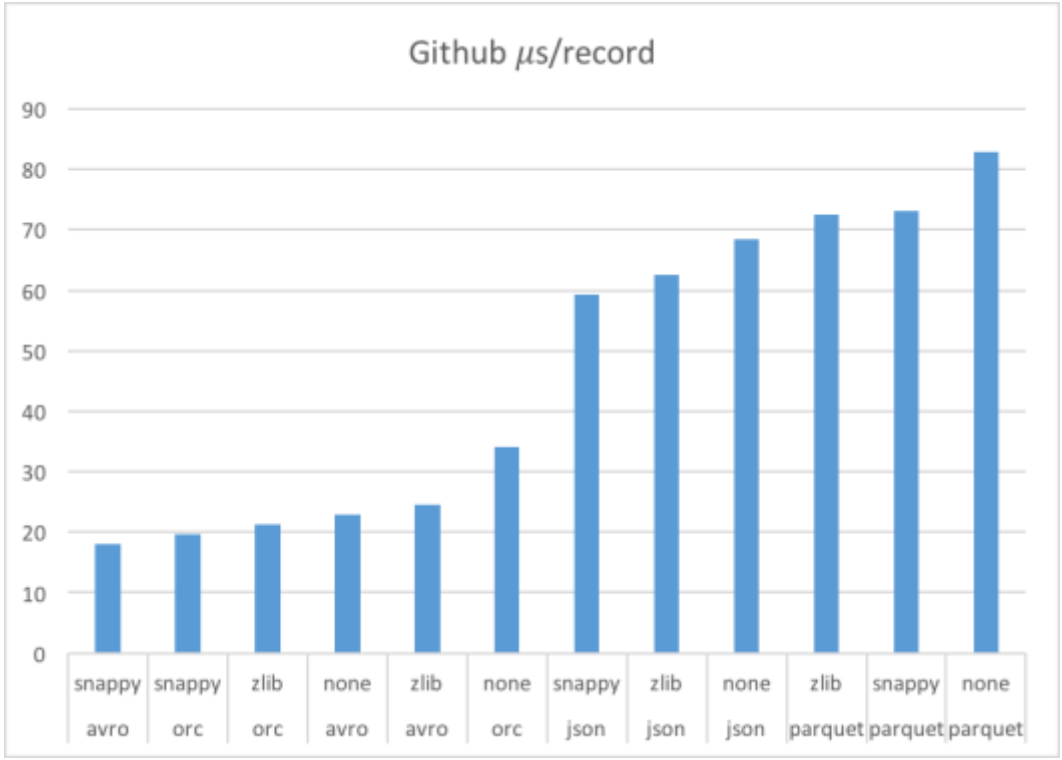
- JSON is very slow to read
  - Large storage size for this data set
  - Needs to do a **LOT** of string parsing
- Tradeoff between space & time
  - Less compression is sometimes faster

HORTONWORKS



Sales Read Performance Analysis

- Read performance is dominated by format
  - Compression matters less for this data set
  - Straight ordering: ORC, Avro, Parquet, & JSON
- Garbage collection is important
  - ORC 0.3 to 1.4% of time
  - Avro < 0.1% of time
  - Parquet 4 to 8% of time



Github Read Performance Analysis

- Garbage collection is critical
  - ORC 2.1 to 3.4% of time
  - Avro 0.1% of time
  - Parquet 11.4 to 12.8% of time
- A lot of columns needs more space
  - Suspect that we need bigger stripes
  - Rows/stripe - ORC: 18.6k, Parquet: 88.1k

HORTONWORKS

Column Projection

- Often just need a few columns
  - Only ORC & Parquet are columnar
  - Only read, decompress, & deserialize some columns

Dataset	format	compression	us/row	projection	Percent time
github	orc	zlib	21.319	0.185	0.87%
github	parquet	zlib	72.494	0.585	0.81%
sales	orc	zlib	1.866	0.056	3.00%
sales	parquet	zlib	12.893	0.329	2.55%
taxi	orc	zlib	2.766	0.063	2.28%
taxi	parquet	zlib	3.496	0.718	20.54%

Projection & Predicate Pushdown

- Sometimes have a filter predicate on table
  - Select a superset of rows that match
  - Selective filters have a huge impact
- Improves data layout options
  - Better than partition pruning with sorting
- ORC has added optional bloom filters

HORTONWORKS

Metadata Access

- ORC & Parquet store metadata
  - Stored in file footer
  - File schema
  - Number of records
  - Min, max, count of each column
- Provides O(1) Access

HORTONWORKS

Conclusions

HORTONWORKS

Recommendations

- Disclaimer – **Everything** changes!
  - Both these benchmarks and the formats will change.
- Don't use JSON for processing.
- If your use case needs column projection or predicate push down:
  - ORC or Parquet

HORTONWORKS

Recommendations

- For complex tables with common strings
  - Avro with Snappy is a good fit (w/o projection)
- For other tables
  - ORC with Zlib or Snappy is a good fit
- Tweet benchmark suggestions to @owen_omalley

HORTONWORKS

Fun Stuff

- Built open benchmark suite for files
- Built pieces of a tool to convert files
  - Avro, CSV, JSON, ORC, & Parquet
- Built a random parameterized generator
  - Easy to model arbitrary tables
  - Can write to Avro, ORC, or Parquet

HORTONWORKS

Remaining work

- Extend benchmark with LZO and LZ4.
- Finish predicate pushdown benchmark.
- Add C++ reader for ORC, Parquet, & Avro.
- Add Presto ORC reader.

HORTONWORKS

Thank you!

Twitter: @owen_omalley
Email: owen@hortonworks.com

HORTONWORKS
POWERING THE FUTURE OF DATA