



ORC Files

Owen O'Malley

owen@hortonworks.com

[@owen_omalley](https://twitter.com/owen_omalley)

June 2013



Who Am I?

- First committer added to Hadoop in 2006
- First VP of Hadoop at Apache
- Was architect of MapReduce
- Was architect of Security
- Broke terasort, minute, and petasort record
- Working on Hive for last 1.5 years
 - Working on Hive code
 - Deep engagement with a large Hive user

History

- Originally Hive used MapReduce's FileFormats
 - Text
 - SequenceFile
- RCFile was developed to improve matters
 - Only read required columns (projection)
 - Each column is stored separately
 - Compresses columns separately
 - Avoid deserializing extra columns
 - select * from people where age > 99
 - Lazy deserialization

Remaining Challenges

- Stores columns as type-free binary blobs
 - Can't optimize representation by type
 - Can't keep meaningful indexes
 - Complex types were not decomposed
- Seeking into file requires byte by byte search
 - Need to find the sync marker
- Compression done using stream-based codecs
 - Decompress column from start of group
- Metadata is added at beginning of file
- Lazy deserialization is very slow

Requirements

- Two Primary Goals
 - Improve query speed
 - Improve storage efficiency
- Support for the Hive type model
 - Complex types (struct, list, map, union)
 - New types (datetime, decimal)
- A single file as output of each task.
 - Dramatically simplifies integration with Hive
 - Lowers pressure on the NameNode
- Bound the amount of memory required

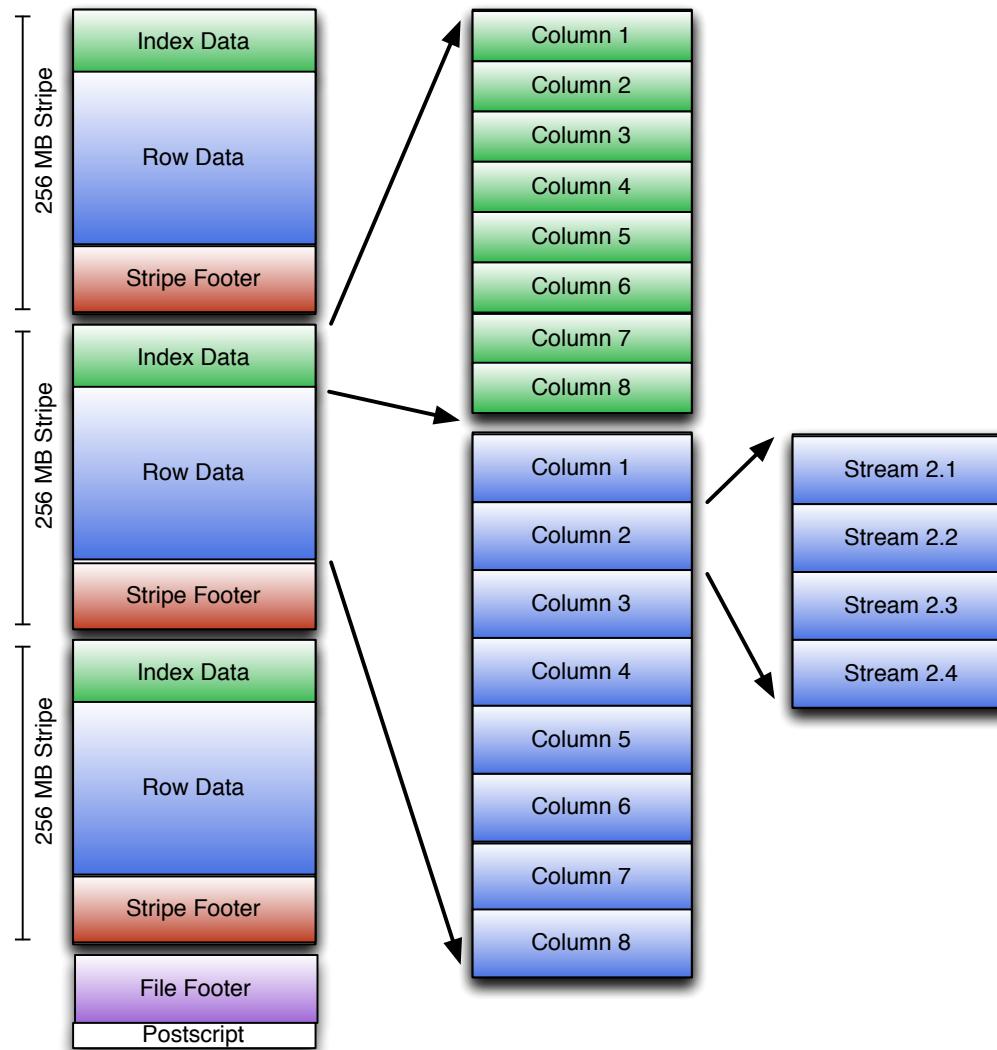
File Structure

- Break file into sets of rows called a stripe.
 - Default stripe size is 256 MB
 - Large size enables efficient read of columns
- Footer
 - Contains list of stripe locations
 - Self-describing types
 - Count, min, max, and sum for each column
- Postscript
 - Contains compression parameters
 - Size of compressed footer

Stripe Structure

- Data
 - Composed of multiple streams per column
- Index
 - Required for skipping rows
 - Defaults to every 10,000 rows
 - Position in each stream
 - Min and max for each column
- Footer
 - Directory of stream locations
 - Encoding of each column

File Layout



Compression

- Light-Weight Compression
 - Type specific (int, string, timestamp, etc.)
 - Always enabled
 - Focus on being very inexpensive
- Generic Compression
 - Support for none, Snappy, LZO, and Zlib
 - Entire file uses the same compression
 - Applied to each stream and footer
 - Applied after light-weight compression

Integer Column Serialization

- Protobuf style variable length integers
 - Small integers have small representations
- Run Length Encoding
 - Uses byte for run length
 - Runs can include very small fixed delta
 - Non-runs use only 1 extra byte / 128 values
- Integer columns have two streams:
 - Present bit stream – is the value non-null?
 - Data stream – stream of integers

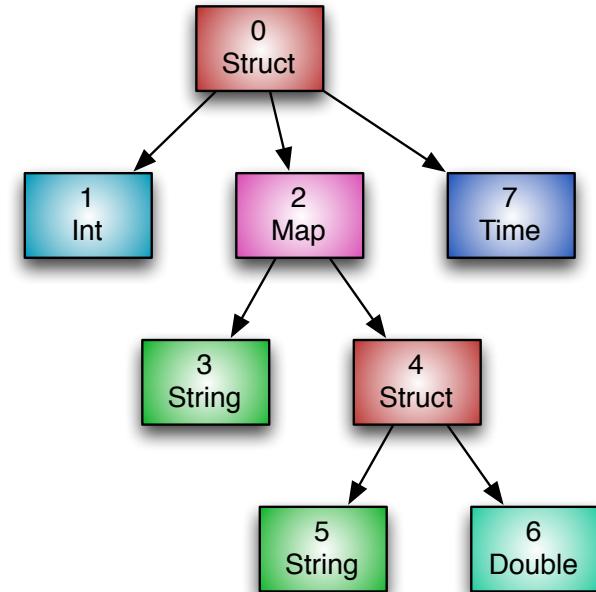
String Column Serialization

- Use a dictionary to uniquify column values
 - Speeds up predicate filtering
 - Improves compression
 - Uses the same run length encoding
- String columns have four streams:
 - Present bit stream – is the value non-null?
 - Dictionary data – the bytes for the strings
 - Dictionary length – the length of each entry
 - Row data – the row values

Hive Compound Types

- Create a tree of readers/writers
- A different class of reader or writer for each type
- Compound types use children for their values

```
create table Tbl (
    col1 int,
    col2 map<string,
        struct<col5:string,
            col6:double>>,
    col7 timestamp)
```



Compound Type Serialization

- Lists
 - Encode the number of items in each value
 - Uses the same run length encoding
 - Uses child writer for value
- Maps
 - Encode the number of items in each value
 - Uses the same run length encoding
 - Uses the key and value child writers

Generic Compression

- Codecs are used to compress blocks of bytes
 - Each compression block is prefixed by a 3 byte header
 - Was the codec used or not?
 - What is the compressed length?
- To record a position in a compressed stream:
 - Offset of the start of compression header
 - Number of uncompressed bytes in block
 - Also the count from run length encoding

Column Projection

- Hive puts the list of required column numbers for the current query in the configuration.
 - Only has ability to request top-level columns
 - Can't request the empty list of columns
- ORC uses the stripe footer to find the requested columns' data streams
 - Reads of adjacent streams are merged
 - Large stripe size means the HDFS reads are reasonable size

How Do You Use ORC

- Added as a storage choice
 - create table Tbl (col int) stored as orc;
- **Do NOT change the Serde!**
 - ORC in/output formats require OrcSerde
- Control table properties
 - orc.compress default ZLIB
 - orc.compress.size default 256K
 - orc.stripe.size default 256MB
 - orc.row.index.stride default 10K

Managing Memory

- The entire stripe needs to be buffered
- ORC uses a memory manager
 - Scales down buffer sizes as number of writers increase
 - Configured using `hive.exec.orc.memory.pool`
 - Defaults to 50% of JVM heap size
- It may improve performance to provide more memory to insert queries.
 - Scaling down stripe size cuts disk IO size
 - Use MapReduce's High RAM jobs

Pavan's Trick

- During data ingest, you often need to write to many dynamic partitions.
 - Pavan was writing to thousands of partitions
 - Even RCFile's small 4MB buffers were OOM
- Re-wrote query as:
from staging
 - insert into partition(part) where key is between x1 and x2
 - insert into partition(part) where key is between x2 and x3
- Reads input once, but splits writing tasks so that each writes to fewer partitions

Looking at ORC File Structures

- A useful tool is orc file dump
- Invoked as “hive –service orcfiledump file.orc”
- Shows file information, statistics, and stripe info

Rows: 21000 **Compression:** ZLIB **Compression size:** 10000

Type: struct<i:int,l:bigint,s:string>

Statistics:

Column 0: count: 21000

Column 1: count: 21000 **min:** -214 **max:** 214 **sum:** 193017

Column 2: count: 21000 **min:** -922 **max:** 922 **sum:** 45060

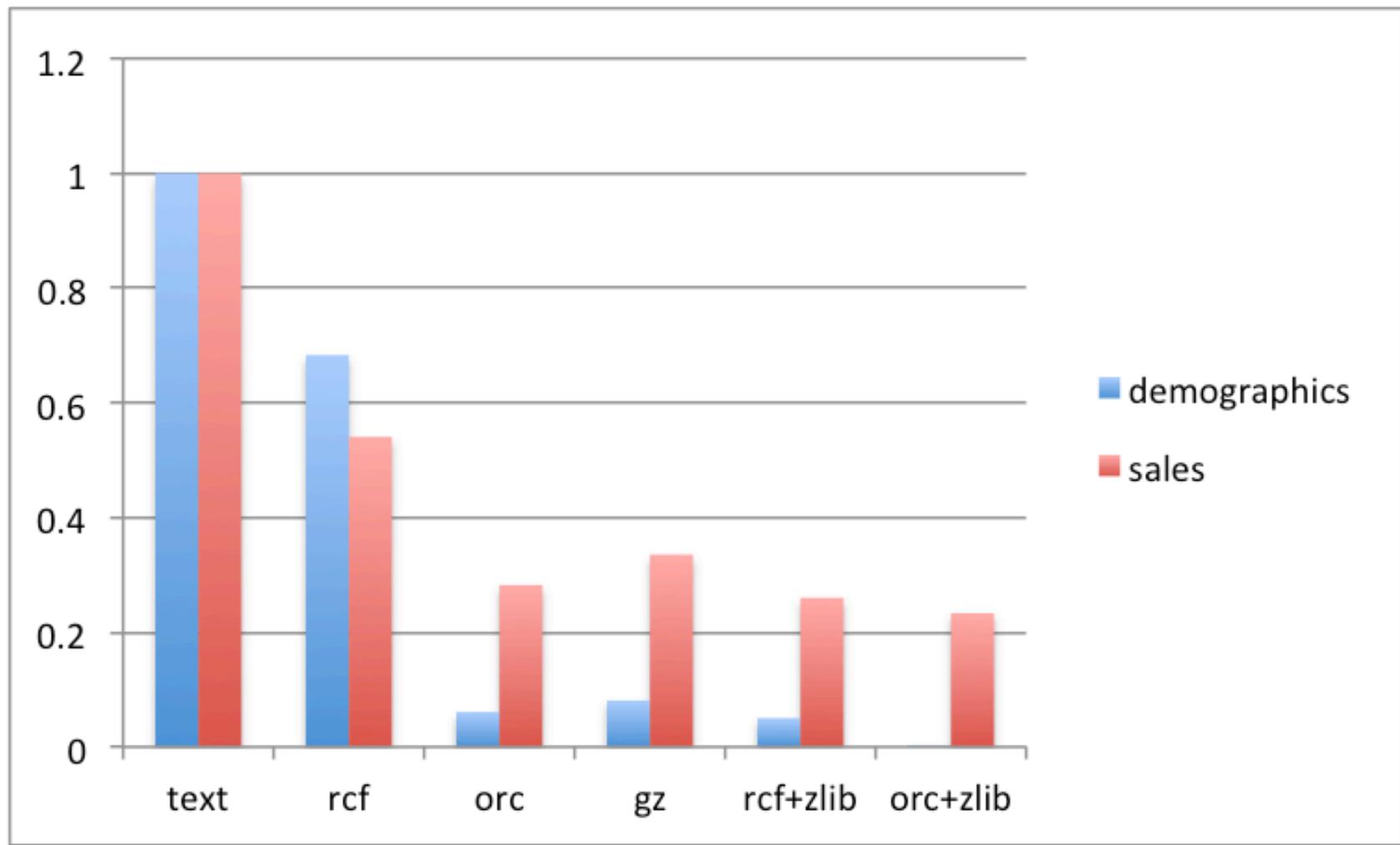
Column 3: count: 21000 **min:** Darkness, **max:** worst

Looking at ORC File Structures

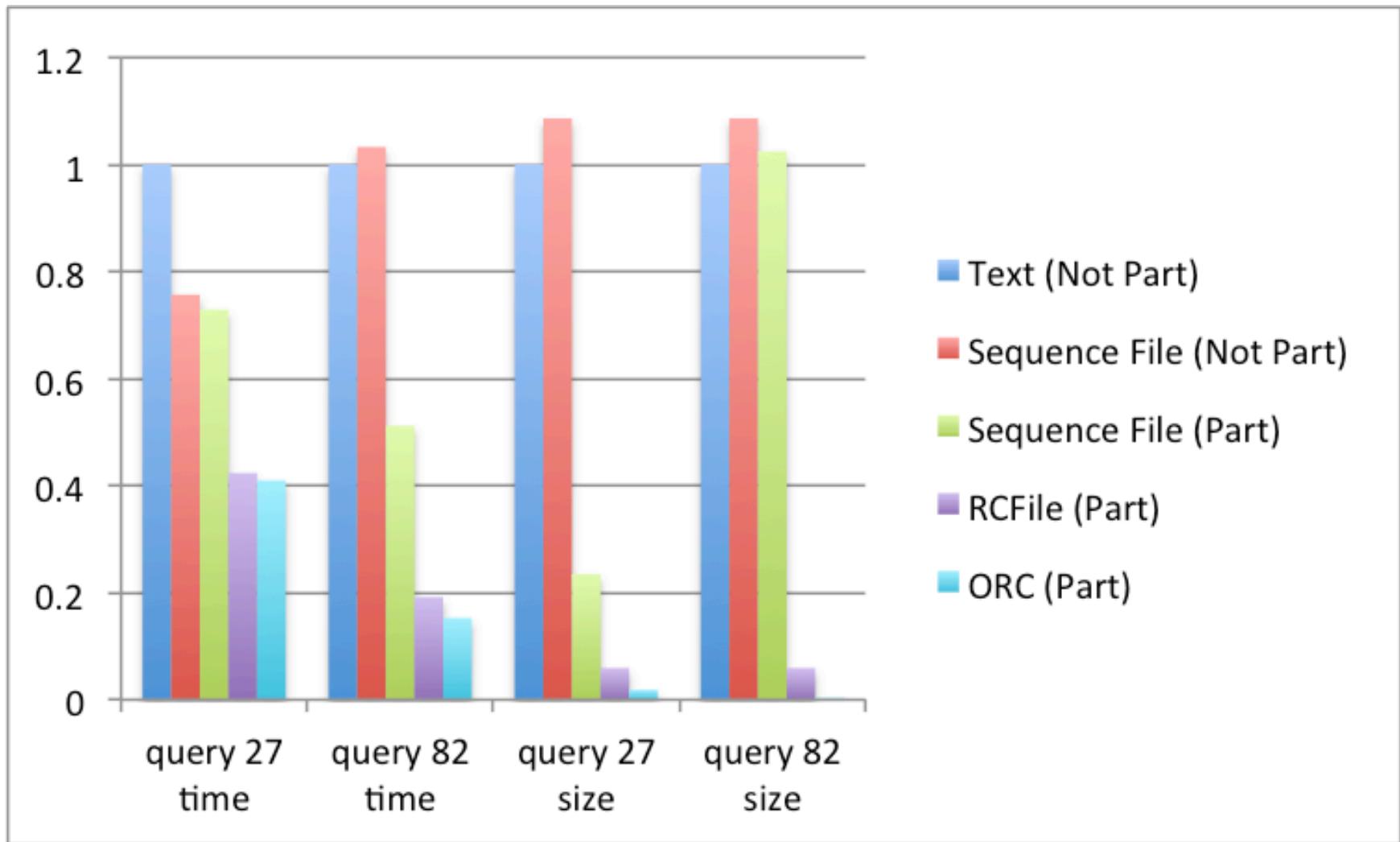
Stripes:

Stripe: offset: 3 data: 69638 rows: 5000 tail: 85 index: 126
column 0 section ROW_INDEX start: 3 length 10
column 1 section ROW_INDEX start: 13 length 38
column 2 section ROW_INDEX start: 51 length 42
column 3 section ROW_INDEX start: 93 length 36
column 1 section PRESENT start: 129 length 11
column 1 section DATA start: 140 length 22605
column 2 section PRESENT start: 22745 length 11
column 2 section DATA start: 22756 length 43426
column 3 section PRESENT start: 66182 length 11
column 3 section DATA start: 66193 length 3403
column 3 section LENGTH start: 69596 length 38
column 3 section DICTIONARY_DATA start: 69634 length 133

TPC-DS File Sizes



TPC-DS Query Performance



Additional Details

- Metadata is stored using Protocol Buffers
 - Allows addition and removal of fields
- Reader supports seeking to a given row #
- ORC doesn't include checksums,
 - Already done in HDFS
- User metadata can be added at any time before the ORC writer is closed.
- ORC files can be read or written from MapReduce or Pig using HCat.

Current work

- Predicate pushdown row group skipping
 - Take subset of where clause and compare to min/max values in index to skip 10k rows
- Make string columns sample the written data and pick their encoding appropriately
 - After 100k values, if dictionary has more than 80k entries use a direct encoding.
- Add optional dictionary encoding of numbers.
- Suppress present stream if no null values
- Improve run length encoding

Vectorization

- Hive uses Object Inspectors to work on a row
 - Enables level of abstraction
 - Costs major performance
 - Exacerbated by using lazy serdes.
- Inner loop has many many method calls
 - Kills performance on modern CPUs
 - Completely trashes the L1 and L2 caches
- Replace inner loop with direct data access
 - Operate on 1000 rows at once
 - Store primitives in primitive arrays

Vectorization Preliminary Results

- First rough prototype was able to run queries in memory at 100 million rows per a second.
- Tasks are getting 10 to 20 times faster
- Involves completely re-writing the engine
- Being developed in a Hive dev branch
- Will have adaptors to switch from vectorized to non-vectorized execution until all of the operators are converted.
- Thanks to Eric Hanson, Jitendra Pandey, Remus Rusanu, Sarvesh Sakalanaga, Teddy Choi, and Tony Murphy

Future Work

- Support count(*), min, max, and average from metadata
- Extend index with optional bloom filter for string columns
- Extend push down predicates with bloom filters
- Allow MapReduce InputSplits to split stripes.
- Writer may reorder rows to improve compression.
 - Must preserve requested sort order

Thanks!

- Owen O'Malley
- owen@hortonworks.com
- [@owen_omalley](https://twitter.com/@owen_omalley)

Comparison

| | RC File | Trevni | Parquet | ORC File |
|-----------------------------|---------|--------|---------|----------|
| Hive Type Model | N | N | N | Y |
| Separate complex columns | N | Y | Y | Y |
| Splits found quickly | N | Y | Y | Y |
| Default column group size | 4MB | 64MB* | 64MB* | 256MB |
| Files per a bucket | 1 | > 1 | 1* | 1 |
| Store min, max, sum, count | N | N | N | Y |
| Versioned metadata | N | Y | Y | Y |
| Run length data encoding | N | N | Y | Y |
| Store strings in dictionary | N | N | N | Y |
| Store row count | N | Y | N | Y |
| Skip compressed blocks | N | N | N | Y |
| Store internal indexes | N | N | N | Y |