Optimizing Hive Queries

Owen O'Malley
Founder and Architect
owen@hortonworks.com
@owen_omalley

Who Am I?

- **Founder and Architect at Hortonworks**
  - Working on Hive, working with customer
  - Formerly Hadoop MapReduce & Security
  - Been working on Hadoop since beginning
- **Apache Hadoop, ASF**
  - Hadoop PMC (Original VP)
  - Tez, Ambari, Giraph PMC
  - Mentor for: Accumulo, Kafka, Knox
  - Apache Member

Outline

- **Data Layout**
- **Data Format**
- **Joins**
- **Debugging**

Data Layout

Location, Location, Location

Fundamental Questions

- **What is your primary use case?**
  - What kind of queries and filters?
- **How do you need to access the data?**
  - What information do you need together?
- **How much data do you have?**
  - What is your year to year growth?
- **How do you get the data?**

HDFS Characteristics

- **Provides Distributed File System**
  - Very high aggregate bandwidth
  - Extreme scalability (up to 100 PB)
  - Self-healing storage
  - Relatively simple to administer
- **Limitations**
  - Can't modify existing files
  - Single writer for each file
  - Heavy bias for large files (> 100 MB)

Choices for Layout

- **Partitions**
  - Top level mechanism for pruning
  - Primary unit for updating tables (& schema)
  - Directory per value of specified column
- **Bucketing**
  - Hashed into a file, good for sampling
  - Controls write parallelism
- **Sort order**
  - The order the data is written within file

Example Hive Layout

- **Directory Structure**

  warehouse/$database/$table

- **Partitioning**

  /part1=$partValue/part2=$partValue

- **Bucketing**

  /$bucket_$attempt (eg. 000000_0)

- **Sort**

  –Each file is sorted within the file

Layout Guidelines

- **Limit the number of partitions**
  - 1,000 partitions is much faster than 10,000
  - Nested partitions are almost always wrong
- **Gauge the number of buckets**
  - Calculate file size and keep big (200-500MB)
  - Don’t forget number of files (Buckets * Parts)
- **Layout related tables the same way**
  - Partition
  - Bucket and sort order

Normalization

- **Most databases suggest normalization**
  - Keep information about each thing together
  - Customer, Sales, Returns, Inventory tables
- **Has lots of good properties, but…**
  - Is typically slow to query
- **Often best to denormalize during load**
  - Write once, read many times
  - Additionally provides snapshots in time.

Data Format

How is your data stored?

Choice of Format

- **Serde**
  - How each record is encoded?
- **Input/Output (aka File) Format**
  - How are the files stored?
- **Primary Choices**
  - Text
  - Sequence File
  - RCFile
  - ORC (Coming Soon!)

Text Format

- **Critical to pick a Serde**
  - Default - ^A’s between fields
  - JSON – top level JSON record
  - CSV – commas between fields (on github)
- **Slow to read and write**
- **Can’t split compressed files**
  - Leads to huge maps
- **Need to read/decompress all fields**

Sequence File

- **Traditional MapReduce binary file format**
  - Stores **keys** and **values** as classes
  - Not a good fit for Hive, which has SQL types
  - Hive always stores entire row as value
- **Splittable but only by searching file**
  - Default block size is 1 MB
- **Need to read and decompress all fields**

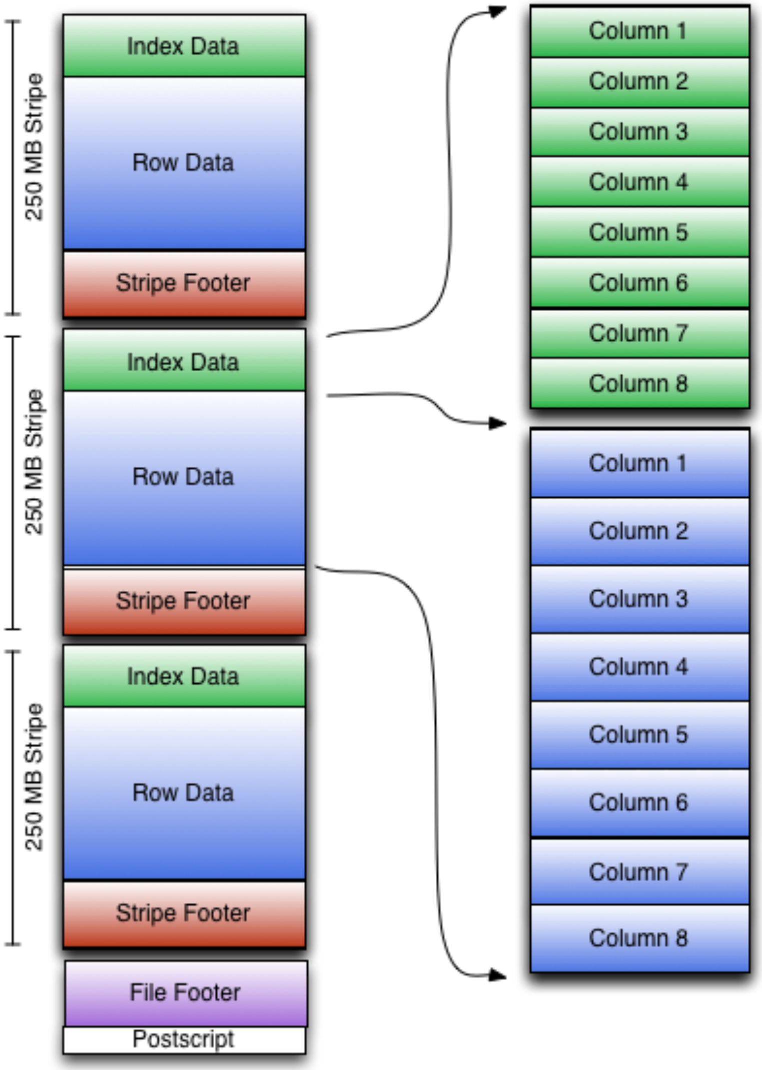
RC (Row Columnar) File

- **Columns stored separately**
  - Read and decompress only needed ones
  - Better compression
- **Columns stored as binary blobs**
  - Depends on metastore to supply types
- **Larger blocks**
  - 4 MB by default
  - Still search file for split boundary

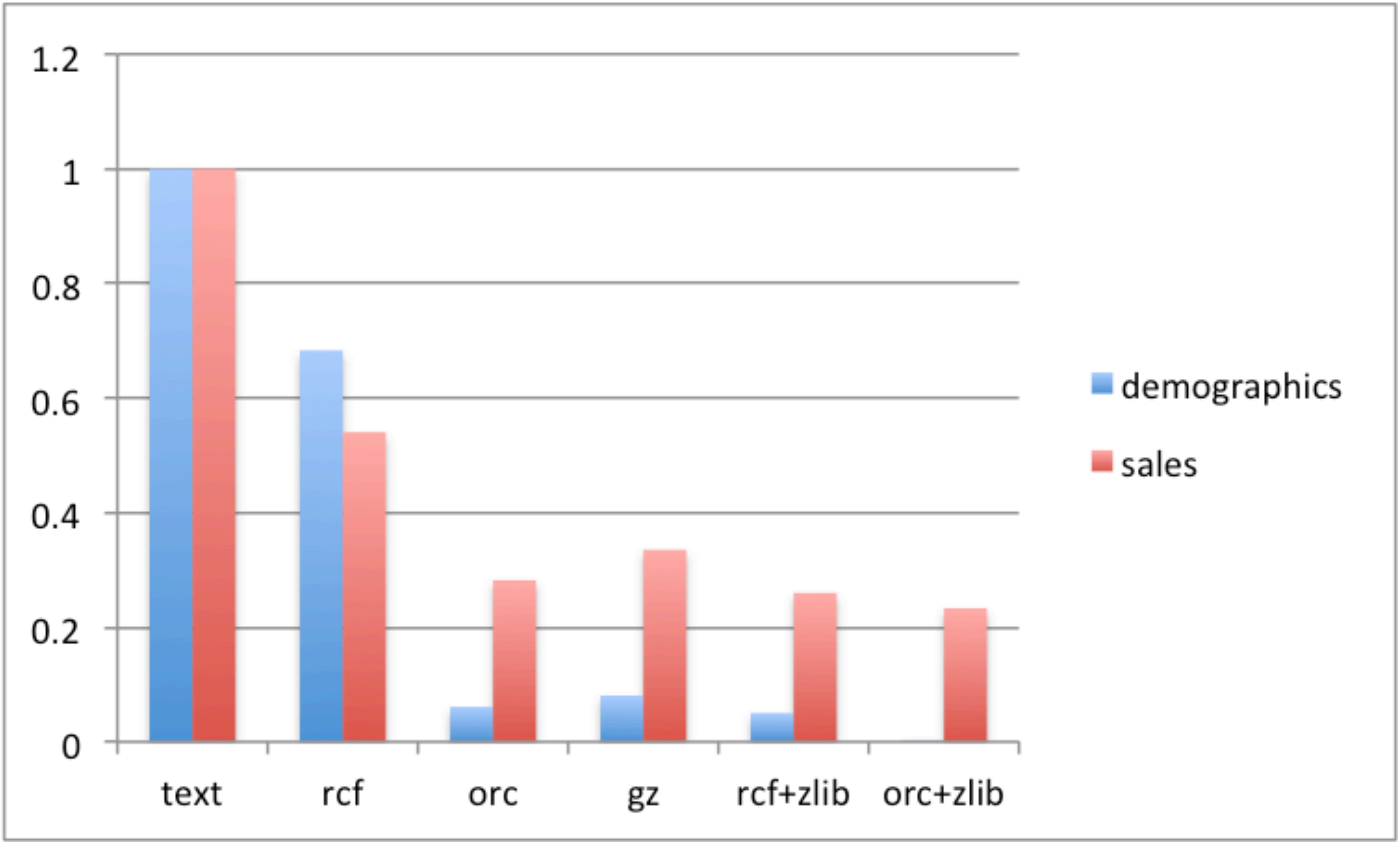
ORC (Optimized Row Columnar)

- **Columns stored separately**
- **Knows types**
  - Uses type-specific encoders
  - Stores statistics (min, max, sum, count)
- **Has light-weight index**
  - Skip over blocks of rows that don't matter
- **Larger blocks**
  - 256 MB by default
  - Has an index for block boundaries

ORC - File Layout



Example File Sizes from TPC-DS



Compression

- **Need to pick level of compression**
  - None
  - LZO or Snappy – fast but sloppy
    - Best for temporary tables
  - ZLIB – slow and complete
    - Best for long term storage

Joins

Putting the pieces together

Default Assumption

- **Hive assumes users are either:**
  - Noobies
  - Hive developers
- **Default behavior is always finish**
  - Little Engine that Could!
- **Experts could override default behaviors**
  - Get better performance, but riskier
- **We’re working on improving heuristics**

Shuffle Join

- **Default choice**
  - Always works (I’ve sorted a petabyte!)
  - Worst case scenario
- **Each process**
  - Reads from part of one of the tables
  - Buckets and sorts on join key
  - Sends one bucket to each reduce
- **Works everytime!**

Map Join

- **One table is small (eg. dimension table)**
  - Fits in memory
- **Each process**
  - Reads small table into memory hash table
  - Streams through part of the big file
  - Joining each record from hash table
- **Very fast, but limited**

Sort Merge Bucket (SMB) Join

- **If both tables are:**
  - Sorted the same
  - Bucketed the same
  - And joining on the sort/bucket column
- **Each process:**
  - Reads a bucket from each table
  - Process the row with the lowest value
- **Very efficient if applicable**

Hortonworks

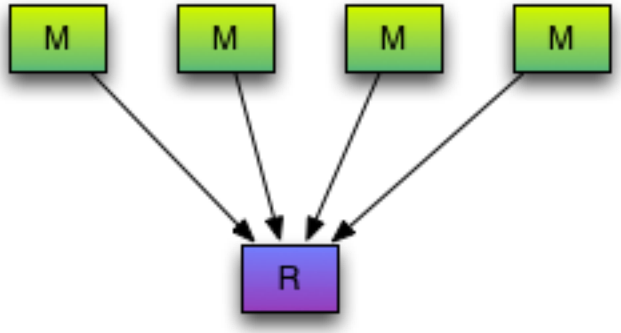
Debugging

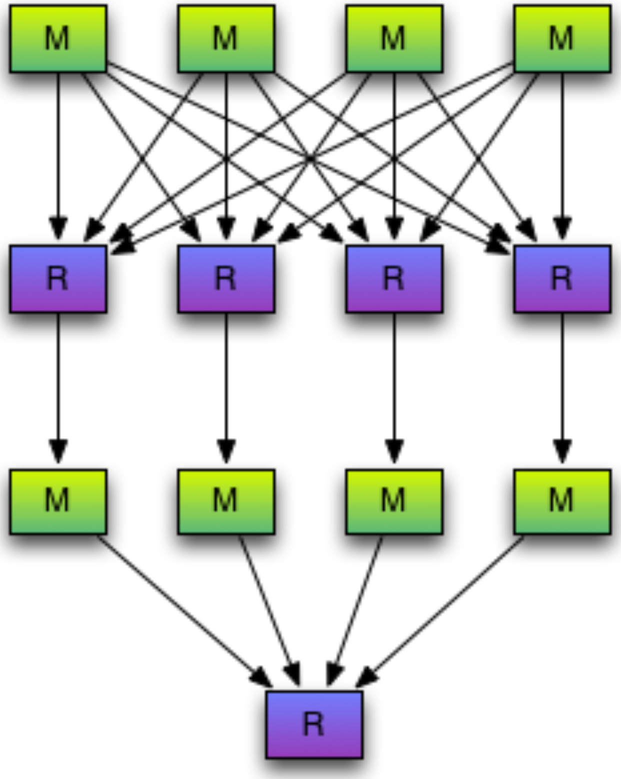
What could possibly go wrong?

Performance Question

- **Which of the following is faster?**
  - select count(distinct(Col)) from Tbl
  - select count(*) from
    (select distict(Col) from Tbl)

Count Distinct





Answer

- **Surprisingly the second is usually faster**
  - In the first case:
    - Maps send each value to the reduce
    - Single reduce counts them all
  - In the second case:
    - Maps split up the values to many reduces
    - Each reduce generates its list
    - Final job counts the size of each list
  - Singleton reduces are almost always BAD

Communication is Good!

- **Hive doesn’t tell you what is wrong.**
  - Expects you to know!
  - “Lucy, you have some ‘splaining to do!”
- **Explain tool provides query plan**
  - Filters on input
  - Numbers of jobs
  - Numbers of maps and reduces
  - What the jobs are sorting by
  - What directories are they reading or writing

Blinded by Science

- **The explanation tool is confusing.**
  - It takes practice to understand.
  - It doesn’t include some critical details like partition pruning.
- **Running the query makes things clearer!**
  - Pay attention to the details
  - Look at JobConf and job history files

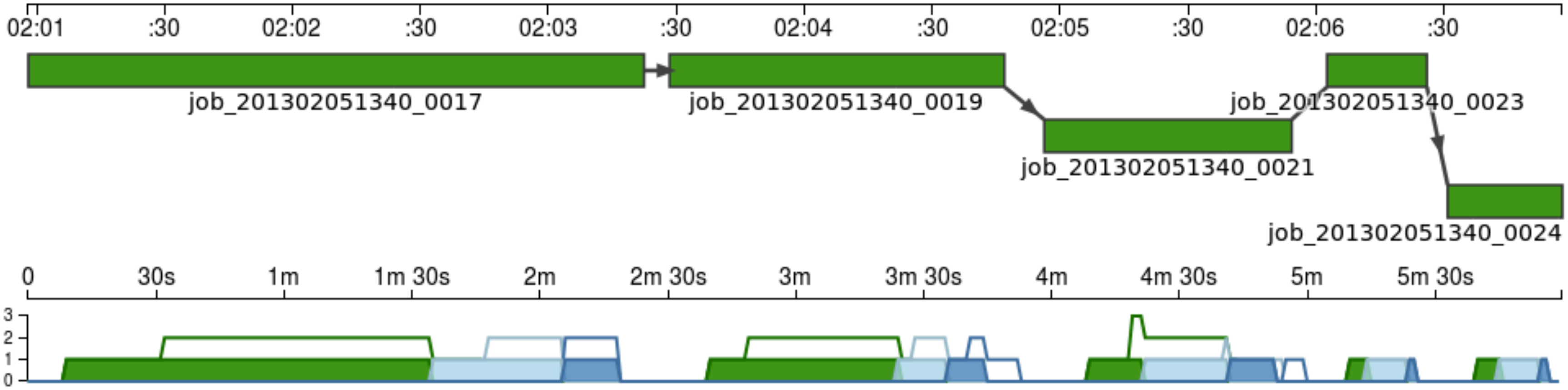
Skew

- **Skew is typical in real datasets.**
- **A user complained that his job was slow**
  - He had 100 reduces
  - 98 of them finished fast
  - 2 ran really slow
- **The key was a boolean…**

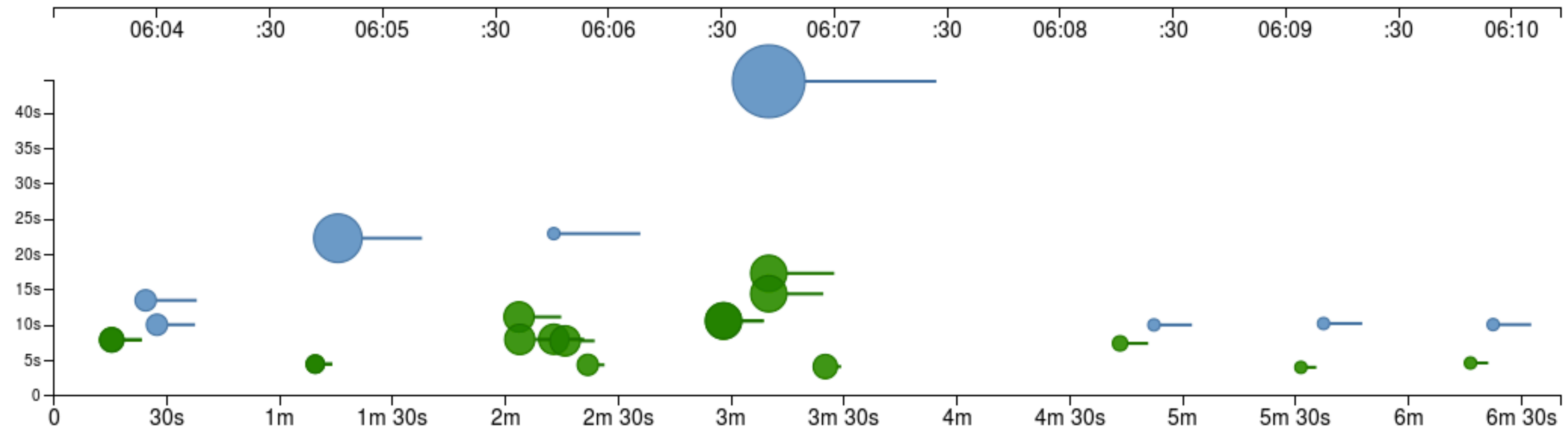
Root Cause Analysis

- **Ambari**
  - Apache project building Hadoop installation and management tool
  - Provides metrics (Ganglia & Nagios)
  - Root Cause Analysis
    - Processes MapReduce job logs
    - Displays timing of each part of query plan

Root Cause Analysis Screenshots



Root Cause Analysis Screenshots



Thank You!

Questions & Answers

@owen_omalley

ORCFile - Comparison

	RC File	Trevni	ORC File
Hive Type Model	N	N	Y
Separate complex columns	N	Y	Y
Splits found quickly	N	Y	Y
Default column group size	4MB	64MB*	250MB
Files per a bucket	1	> 1	1
Store min, max, sum, count	N	N	Y
Versioned metadata	N	Y	Y
Run length data encoding	N	N	Y
Store strings in dictionary	N	N	Y
Store row count	N	Y	Y
Skip compressed blocks	N	N	Y
Store internal indexes	N	N	Y