DATA+AI SUMMIT
BY databricks

DELTA MERGE OPTIMIZATIONS WITH JODIE HELPERS

https://github.com/MrPowers/jodie

Joydeep Banik Roy
Head of Data Science, Zeotap

BIT ABOUT ZEOTAP

AND WHAT WE DO

ZEOTAP CDP

- Integrate, Unify, Segment and Orchestrate customer data for brands
- Intuitive UI for marketers
- Drive business outcomes for your 1st Party data
- Navigate the Cookieless future

ZEOTAP DATA

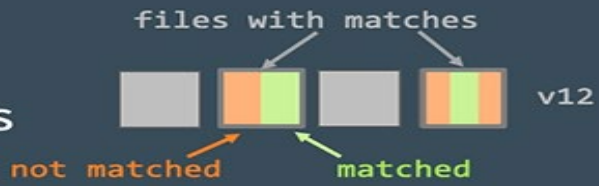
- 3rd Party Aggregated Data
- Consented, GDPR Compliant
- People based, deterministic
- Exclusive Telco data partnerships
- Demographic, IAB 1.1 and other attributes

How Delta Merge Works

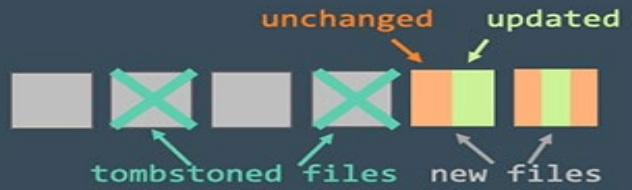
Under the Hood

Merge – Under the hood

Scan 1: Inner join between target and source to select files that have matches



Scan 2: Outer join between the selected files in target and source and write the update/deleted/inserted data



HOW DELTA MERGE WORKS

MERGE WITH UPSERTS

- APPLY DATA SKIPPING
- FIRST JOIN - INNER
  - *notMatchedBySource* Clause - RIGHT OUTER
  - DOES SANITY
    - MULTIPLE SOURCE ROWS NOT MATCHING SINGLE TARGET ROW
  - RECORDS STATS
    - `numTargetFilesBeforeSkipping`
    - `numTargetFilesAfterSkipping`
- WRITING PHASE
  - OUTER JOIN
    - RIGHT OUTER - merge without 'when not matched' clause is optimised
    - FULL OUTER
  - IF CHANGE DATA CAPTURE ENABLED
    - RECORD CDF
  - RECORD STATS
    - `numTargetRowsUpdated`
    - `numTargetRowsNotMatchedBySourceUpdated`
    - `numTargetRowsInserted`
  - WRITE FILES AND RECORD TRANSACTION/VERSION

DATA SKIPPING

- Collecting min max values and null counts on your columns
- Total records per file
- Filter files from Delta Table based on these metrics
  - Candidate files for shuffle

STRATEGIES TO OPTIMIZE

HOW CAN JODIE HELP

INPUT DATA

- FILE STATISTICS
- COMPACTION & VACUUM
- ZORDER AND LIQUID CLUSTERING
- MIN-MAX RANGE

ADVANTAGES

- PULLS OUT THE DATA SKIPPING PART
- NO DATA IS ACTUALLY PULLED INTO MEMORY
    - GIVES A DRY RUN CAPABILITY

```
DeltaHelpers.getNumShuffleFiles
```

DELTA MERGE

File Skipping

Scala

```
# Call with your partition condition
DeltaHelpers.getNumShuffleFiles(path, "country = 'GBR' and age >= 30 and age <= 40 and firstname like '%Jo%' ")

# Returns a MAP with following keys
# +-----------------------------------+-----------------------+---------------------------------------------+
# | MAP KEY                                                  |          Count                                 |
Explanation                                                                |
# +-----------------------------------+-----------------------+---------------------------------------------+
# | OVERALL RESOLVED CONDITION                  |          18                                |
                                                                                                |
# | GREATER THAN / LESS THAN PART    |          100                          |
                                                                                 |
# | EQUALS/EQUALS NULL SAFE PART     |          300                          |
                                                                                 |
# | LEFT OVER PART                                  |          600                               |
                                                                                                |
# | UNRESOLVED PART                                  |          800                               |
                                                                                                |
# | TOTAL_NUM_FILES_IN_DELTA_TABLE  |          800                          |
```

FILE STATISTICS

Class : DeltaHelpers

```
deltaNumRecordDistribution(path,
Some("country='Australia'"))
deltaFileSizeDistribution(path,
Some("country='Australia'"))
deltaFileSizeDistributionInMB(path
, Some("country='Australia'"))
```

Prints as a DF

- No. of Parquet Files
- Mean
  - Num Records in Files
  - Size of Files
- Standard Deviation
- Minimum & Maximum
  - Number of Records
  - File Size
- 10th-95th Percentile

INSERT ONLY MERGE

STEP BY STEP

- APPLY FILE SKIPPING
- FIRST JOIN - LEFT ANTI
  - Figures out only insert candidates
- IF CDC IS ENABLED
  - WRITE CDC WITH ONLY INSERT
- WRITE INSERTS AND RECORD VERSION

INSERT ONLY MERGE

HELPFUL FOR INSERTS

OPTIMIZATION

- HAVE NO MERGE AT ALL
  - USE APPEND WITH DELTA TABLE
    - BLIND APPENDS
- USE WHEN NOT MATCHED WITH ONLY INSERTS
  - LEFT ANTI ENSURES NO DUPLICATE APPEND

ADVANTAGES

- ELIMINATE THOSE EXTRA JOINS
- YOUR UPDATE DF GETS SMALLER WHICH MEANS SMALLER INNER JOIN

MULTI CLUSTER WRITES AT PARTITION LEVEL

STRATEGIES TO OPTIMIZE

CONCURRENCY

- ***their likelihood of working with the exact same data/row would be very low.***
- KEY IDEA : PARTITIONS
    - Boundaries where rows don’t overlap

WHY?

- SPEED OF EXECUTION
    - One large join is now split into multiple concurrent yet smaller joins
- AVOID FAILURES
    - WHEN LARGE MERGE JOB FAILS DUE TO ONE PARTITION, IT PREVENTS DATA TO BE COMMITTED TO OTHER PARTITIONS

DELTA MERGE

Concurrently firing Merges based on Partition

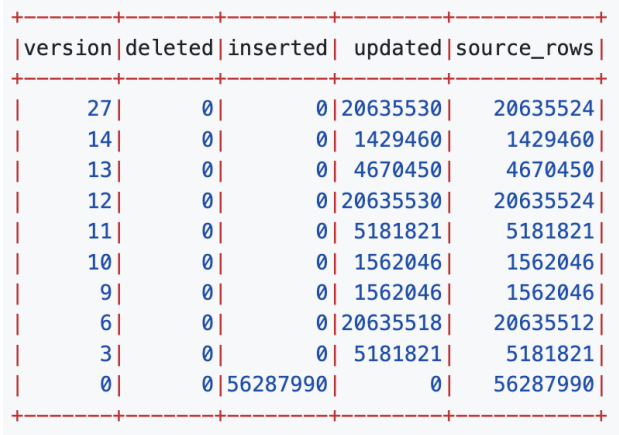
Scala

```
# Call with your partition condition
// Tested on Delta Lake v2.1.0
val df = spark.read.parquet("gs://changeSetPath/country=USA")
         .withColumn("country",lit("USA"))
deltaTable.as("target")
 .merge(df.as("source"),
 //Earlier would have looked like "target.id = source.id and target.country = source.country"
   "target.id = source.id and target.country = 'USA'")
 .whenMatched
 .updateAll()
 .whenNotMatched()
 .insertAll()
 .execute()
```

HOW CAN JODIE HELP

Class : OperationMetricHelper

```
getCountMetricsAsDF()
.show()
getCountMetricsAsDF(
  Some(" country = 'USA' and
gender = 'Female'"))
.show()
```



version	deleted	inserted	updated	source_rows
27	0	0	20635530	20635524
14	0	0	1429460	1429460
13	0	0	4670450	4670450
12	0	0	20635530	20635524
11	0	0	5181821	5181821
10	0	0	1562046	1562046
9	0	0	1562046	1562046
6	0	0	20635518	20635512
3	0	0	5181821	5181821
0	0	56287990	0	56287990

- Comprehensive view of all count metric of a Delta Table
- Portrays Table Growth by showing partition-wise insert, update and delete count

YOU MIGHT BE SURPRISED

We ran this method on our production tables

```
OperationMetricHelper("gs://deltaTablePath").getCountMetricsAsDF()
```

version	deleted	inserted	updated	source_rows
89	1	0	0	23312
88	0	184486	1356673	1541159
87	2	0	0	23311
86	0	147493	1089097	1236590
85	3	0	0	23310
84	0	161111	1370505	1531616
83	2	0	0	23309
82	0	162635	1329323	1491958
81	0	0	0	23309
80	0	127567	1066790	1194357
79	0	0	0	23309
78	0	119795	953362	1073157
77	1	0	0	23309
76	0	178159	1285403	1463562
75	0	0	0	23309
74	0	138243	1040616	1178859
73	0	0	0	23309
72	0	186089	1319314	1505403
71	0	0	0	23299
70	0	151824	1031102	1182926

- ISSUE :
  - versions **71, 73,** and **75** ran without any overlap
  - **MERGE-DELETE** operation that ran at a regular frequency
  - It did the inner join for merge and then created a new version — just to do a **No-Op**
- **FIX :** We reduced its run frequency and merged this operation with others

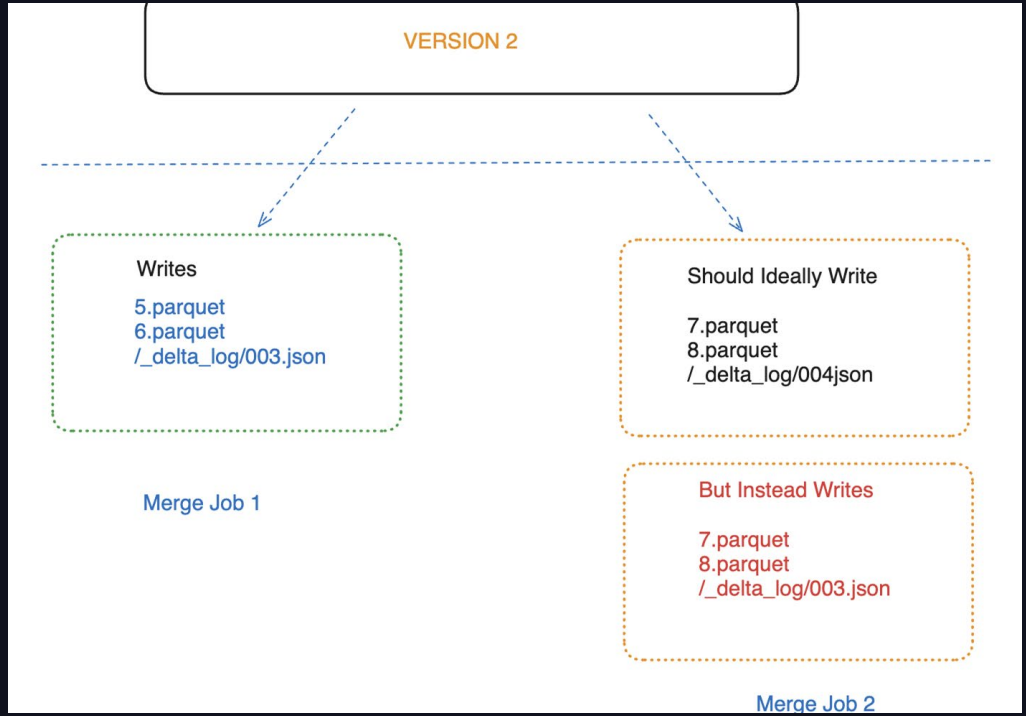
DELETION VECTORS

Multi Cluster Write

Problem

https://delta.io/blog/2022-05-18-multi-cluster-writes-to-delta-lake-storage-in-s3/

- Requires Strong Consistency Guarantees on your Object Storage
- Nothing to do with data ✅
- Affects Transaction Logs ❌
- GCS
  - Missing LogStore impl with Delta 1.1.0
  - https://docs.delta.io/latest/delta-storage.html
  - https://github.com/MrPowers/jodie/pull/83



LOW SHUFFLE MERGE

With or Without PHOTON™

- Optimizes the processing of unmodified rows. In Normal Merge, they were processed in the same way as modified rows, passing them through multiple shuffle stages and expensive calculations. **In low shuffle merge, the unmodified rows are instead processed without any shuffles, expensive processing, or other added overhead.**
- It's not available on Delta Lake OSS
- Trick: **If you have fewer records coming in for upserts, say 0.5 or 1% of the total number of records , use Low Shuffle Merge on DBR else use Delta OSS Merge.**
- Very easy to use both Databricks Runtime and Delta OSS on the same Delta Table

HOW DELTA MERGE WORKS

MERGE WITH UPSERTS

- APPLY DATA SKIPPING
- FIRST JOIN - INNER
  - *notMatchedBySource* Clause - RIGHT OUTER
  - DOES SANITY
    - MULTIPLE SOURCE ROWS NOT MATCHING SINGLE TARGET ROW
  - RECORDS STATS
    - `numTargetFilesBeforeSkipping`
    - `numTargetFilesAfterSkipping`
- WRITING PHASE
  - OUTER JOIN
    - RIGHT OUTER - merge without 'when not matched' clause is optimised
    - FULL OUTER
  - IF CHANGE DATA CAPTURE ENABLED
    - RECORD CDF
  - RECORD STATS
    - `numTargetRowsUpdated`
    - `numTargetRowsNotMatchedBySourceUpdated`
    - `numTargetRowsInserted`
  - WRITE FILES AND RECORD TRANSACTION/VERSION

CHANGE DATA FEED

TREAD CAREFULLY

- Has a huge overhead on your merge writes
- Use jodie helpers to switch them on and off between versions efficiently
- Enable-Disable-Re Enable - Programmatically
    - https://medium.com/@joydeep.roy/change-data-feed-time-travel-failure-scenarios-prevention-recovery-5606c65d0c2e#7fd1
- Example:
    - Understand pattern of data
    - If a huge CDF comes on a particular day of the week, switch it off
    - Depends on your use case and what kind of control you want

FINALLY

HOW TO CONTRIBUTE TO JODIE?

SHOW SOME SCALA 🥰

- Optimizations
- User features
  - Loads of them on jodie already
  - Also checkout mack
    - https://github.com/MrPowers/mack
  - github.com/MrPowers/jodie
  - medium.com/@joydeep.roy

GET ACTIVE ON SLACK

- #deltalake-questions
- Github Issues of
- Delta OSS
- JODIE

THANK YOU

- MATTHEW POWERS
- BRAYAN JACQUES JOULES
- SAI ANIRUDH
- YATHARTH MAHESHWARI
- WARIS CHUTANI

THANK YOU