Hive Does ACID

Owen O'Malley
owen@hortonworks.com
@owen_omalley

September 2015

History

- **Hive only updated partitions**
  - INSERT...OVERWRITE rewrote an entire partition
  - Forced daily or even hourly partitions
  - Could add files to partition directory, but no file compaction
- **What about concurrent readers?**
  - Ok for inserts, but overwrite caused races
  - There is a zookeeper lock manager, but…
- **No way to delete or update rows**
- **No INSERT INTO T VALUES…**
  - Breaks some tools

Why is ACID Critical?

- **Hadoop and Hive have always…**
  - Worked without ACID
  - Perceived as tradeoff for performance
- **But, your data isn't static**
  - It changes daily, hourly, or faster
  - Ad hoc solutions require a lot of work
  - Managing change makes the user's life better

Use Cases

- **NOT OLTP!!!**
- **Updating a Dimension Table**
  - Changing a customer's address
- **Delete Old Records**
  - Remove records for compliance
- **Update/Restate Large Fact Tables**
  - Fix problems after they are in the warehouse
- **Streaming Data Ingest**
  - A continual stream of data coming in
  - Typically from Flume or Storm
- **NOT OLTP!!!**

New SQL in Hive 0.14

- **New DML**
  - INSERT INTO T VALUES(1, ‘fred’, ...);
  - UPDATE T SET (x = 5[, ...]) WHERE ...
  - DELETE FROM T WHERE ...
  - Supports partitioned and non-partitioned tables, WHERE clause can specify partition but not required
- **Restrictions**
  - Table must have format that extends AcidInputFormat
    - currently ORC
    - work started on Parquet (HIVE-8123)
  - Table must be bucketed and not sorted
    - can use 1 bucket but this will restrict write parallelism
  - Table must be marked transactional
    - create table T(...) clustered by (a) into 2 buckets stored as orc TBLPROPERTIES ('transactional'='true');

Design

- **HDFS Does Not Allow Arbitrary Writes**
  - Store changes as delta files
  - Stitched together by client on read
- **Writes get a Transaction ID**
  - Sequentially assigned by Metastore
- **Reads get Committed Transactions**
  - Provides snapshot consistency
  - No locks required
  - Provide a snapshot of data from start of query

Why Not HBase?

- **Good**
  - Handles compactions for us
  - Already has similar data model with LSM
- **Bad**
  - No cross row transactions
    - Would require us to write a transaction manager over HBase, doable, but not less work
  - Hfile is column family based rather than columnar
  - HBase focused on point lookups and range scans
    - Warehousing requires full scans

Stitching Buckets Together

Base File

Name	Purchase
Anne	Red Fish
Bill	Blue Fish
Christine	Blue Fish
David	Black Fish
Eric	Young Fish

Update 1

Op	Txn Id	RowId	Name	Purchase
I	1	0	Joe	Red Fish
U	0	0	Anne	Star
D	0	4		

Update 2

Op	Txn Id	RowId	Name	Purchase
U	1	0	Joe	Old Fish
U	0	0	Ann	Star
D	0	2		

Logical File

Name	Purchase
Joe	Old Fish
Ann	Star
Bill	Blue Fish
David	Black Fish

HDFS Layout

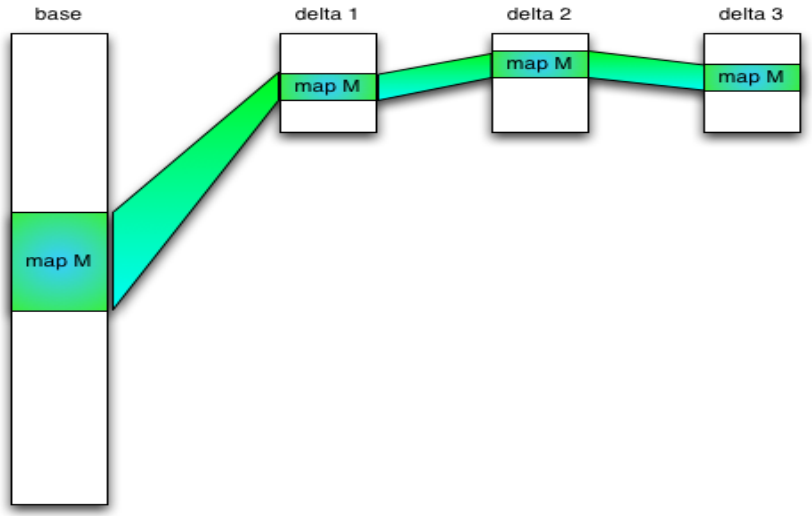
- **Partition locations remain unchanged**
  - Still warehouse/$db/$tbl/$part
- **Bucket Files Structured By Transactions**
  - Base files $part/base_$tid/bucket_*
  - Delta files $part/delta_$tid_$tid/bucket_*

Input and Output Formats

- **Created new AcidInput/OutputFormat**
  - Unique key is transaction, bucket, row
- **Reader returns correct version of row based on transaction state**
- **Also Added Raw API for Compactor**
  - Provides previous events as well
- **ORC implements new API**
  - Extends records with change metadata
    - Add operation (d, u, i), transaction and key

Distributing the Work

- **Need to split buckets for MapReduce**
  - Need to split base and deltas the same way
  - Use key ranges
  - Use indexes



Transaction Manager

- **Existing lock managers**
  - In memory - not durable
  - ZooKeeper - requires additional components to install, administer, etc.
- **Locks need to be integrated with transactions**
  - commit/rollback must atomically release locks
- **We sort of have this database lying around which has ACID characteristics (metastore)**
- **Transactions and locks stored in metastore**
- **Uses metastore DB to provide unique, ascending ids for transactions and locks**

Transaction Model

- **In Hive 0.14 DML statements are auto-commit**
  - Working on adding BEGIN, COMMIT, ROLLBACK
- **Snapshot isolation**
  - Reader will see consistent data for the duration of his/her query
  - May extend to other isolation levels in the future
- **Current transactions can be displayed using new SHOW TRANSACTIONS statement**

Locking Model

- **Three types of locks**
  - shared
  - semi-shared (can co-exist with shared, but not other semi-shared)
  - exclusive
- **Operations require different locks**
  - SELECT, INSERT – shared
  - UPDATE, DELETE – semi-shared
  - DROP, INSERT OVERWRITE – exclusive

Compactor

- **Each transaction (or batch of transactions in streaming ingest) creates a new delta file**
- **Too many files = NameNode ☹**
- **Need a way to**
  - Collect many deltas into one delta – *minor compaction*
  - Rewrite base and delta to new base – *major compaction*

Minor Compaction

- **Run when there are 10 or more deltas (configurable)**
- **Results in base + 1 delta**

```
/hive/warehouse/purchaselog/ds=201403311000/base_0028000
/hive/warehouse/purchaselog/ds=201403311000/delta_0028001_0028100
/hive/warehouse/purchaselog/ds=201403311000/delta_0028101_0028200
/hive/warehouse/purchaselog/ds=201403311000/delta_0028201_0028300
/hive/warehouse/purchaselog/ds=201403311000/delta_0028301_0028400
/hive/warehouse/purchaselog/ds=201403311000/delta_0028401_0028500
```

```
/hive/warehouse/purchaselog/ds=201403311000/base_0028000
/hive/warehouse/purchaselog/ds=201403311000/delta_0028001_0028500
```

Major Compaction

- **Run when deltas are 10% the size of base (configurable)**
- **Results in new base**

```
/hive/warehouse/purchaselog/ds=201403311000/base_0028000
/hive/warehouse/purchaselog/ds=201403311000/delta_0028001_0028100
/hive/warehouse/purchaselog/ds=201403311000/delta_0028101_0028200
/hive/warehouse/purchaselog/ds=201403311000/delta_0028201_0028300
/hive/warehouse/purchaselog/ds=201403311000/delta_0028301_0028400
/hive/warehouse/purchaselog/ds=201403311000/delta_0028401_0028500
```

```
/hive/warehouse/purchaselog/ds=201403311000/base_0028500
```

Compactor Continued

- **Metastore thrift server will schedule and execute compactions**
  - No need for user to schedule
  - User can initiate via ALTER TABLE COMPACT statement
- **No locking required, compactions run at same time as select and DML**
  - Compactor aware of readers, does not remove old files until readers have finished with them
- **Current compactions can be viewed using new SHOW COMPACTIONS statement**

Application: Streaming Ingest

- **Data is flowing in from generators in a stream**
- **Without this, you have to add it to Hive in batches, often every hour**
  - Thus your users have to wait an hour before they can see their data
- **New interface in hive.hcatalog.streaming lets applications write small batches of records and commit them**
  - Users can now see data within a few seconds of it arriving from the data generators
- **Available for Apache Flume in HDP 2.1 and Storm in HDP 2.2**

Phases of Development

- **Hive 0.13**
  - Transaction and new lock manager
  - ORC file support
  - Automatic and manual compaction
  - Snapshot isolation
  - Streaming ingest via Flume
- **Hive 0.14**
  - INSERT … VALUES, UPDATE, DELETE
- **Hive 1.2**
  - Add support for only some columns in insert (HIVE-9481, will be in Hive 1.2)
    - INSERT into T (a, b) select c, d from U;
- **Future (all speculative based on user feedback)**
  - MERGE
  - Integration with HCatalog
  - Versioned or point in time queries
  - Additional isolation levels such as dirty read or read committed

Current Work

- **Multi-statement transactions**
  - BEGIN, COMMIT, ROLLBACK (HIVE-9675)
- **Add support for update/delete in streaming ingest (HIVE-10165)**
- **Add support for Parquet files (HIVE-8123)**

Next: MERGE

- **Standard SQL, added 2003**
- **Allows upserts**
- **Use case:**
  - bring in batch from transactinal/front end systesm
  - Apply as insert or updates (as appropriate) in one read/write pass

What Next?

- **Better performance**
  - In Tez not doing split combination efficiently in delta only case (fixed in Tez 0.7)
  - Predicate push down not applied to deltas
- **Better usability**
  - Remove requirements for bucketing
  - Remove need to mark table transactional

Conclusion

- **JIRA: HIVE-5317**
- **Adds ACID semantics to Hive**
- **Uses SQL standard commands**
  - INSERT, UPDATE, DELETE
- **Provides scalable read and write access**

Thank You!

Owen O'Maley

@owen_omalley

owen@hortonworks.com