2018/12/19 - Eugene Koifman

Issue

When Streaming Ingest V2 or SQL Insert into Micro Managed table is run, with dynamic partitioning (DP), transaction failure may result in aborted data being returned in queries.

The basic design of Hive ACID tables is that all data written to disk is tagged with a transaction id (via write id) so that at read time any data that belongs to Aborted/Open transaction can be ignored by the reader. For this to work we need keep a record about the aborted txn until the compactor has had a chance to make sure that all data written by this txn has been physically removed from disk. In practice this means the Compactor has to visit every partition written by the aborted transaction. Thus before any data in a given partition hits the disk, we need to save metadata in the Acid system to indicate what partitions a txn touched. Specifically, it means update TXN_COMPONENTS table in the HMS. Once all data written by an aborted transaction is physically removed, the metadata about this transaction can be purged.

With DP, we don't know what partitions are written until they are written by the tasks, especially since the write may create new partitions. The way this is handled for full CRUD tables, is by making a call in Acid system (addDynamicPartitions()) from MoveTask once the set of partitions Written/created is known but before new deltas are moved from temp location to the table directory. (V1 streaming ingest doesn't support DP so it knows what partitions it is writing from the start.)

The same doesn't work for DP with MM and V2 streaming since both create deltas in the table dir from the start (i.e. MoveTask doesn't do FileSystem.rename()), so if a txn fails before addDynamicPartitions() is called, Acid system removes the metadata about the txn before all data written by it is removed so this data starts to look like committed data.

Solution

When a transaction is performing an operation with DP, create an entry in TXN_COMPONENTS with TC_OPERATION_TYPE='p' (this is a new type. TxnHandler.OperationType) and TC_PARTITION = NULL. This should happen before any data is written to disk. When addDynamicPartitions() callback is made, this entry should be deleted and replaced with actual list of partitions in this table. For committed transactions or those that fail after addDynamicPartitions() is done, the rest of the flow is exactly as today.

If the transaction fails before addDynamicPartitions(), then the p-type record in TXN_COMPONENTS will prevent the metadata about the transaction (in TXNS table) from being deleted until TXN_COMPONENTS entry is removed (modify markCleaned() for this to work). In order to remove this TXN_COMPONENTS entry, we have to examine every partition in the table and remove any [delete_]delta_writeId_writeId_stmtId. Both table name and writeId are part of the TXN_COMPONENTS entry. (More precisely, examine the whole file tree of the table - failed operation may have written to partitions that don't exist yet).

NOTE: this only works for Streaming Ingest if TransactionBatch size = 1. If it's greater than 1, actual compaction needs to run which is prohibitively expensive to do on all partitions of a large table. Furthermore, since the failed insert may have written data to partitions that didn't exist at the time, (i.e. addPartition HMS operation didn't take place before the failure), it's physically impossible to run compactor on a non-existent partition. So this should be asserted.

Implementation Notes

The Initiator should special case the p-type records and add them to the COMPACTION_QUEUE directly in the ready-for-cleaning state. This way all the monitoring (like SHOW COMPACTIONS) still captures these. COMPACTION_QUEUE.CQ_TYPE should have a new type, something like clean-aborted in addition to major/minor. CQ_RUN_AS should also be set by Initiator in this case. TxnStore.markCleaned() should have special case for p-type records, i.e. it either deletes TXN_COMPONENTS rows for clean-aborted type or major/minor. This is to make sure that if we have aborted DP transactions mixed with other aborted txns, the 'other' txn clean doesn't remove the p-type record before all partitions affected by it are handled. Basically, the "select distinct txn_id from TXNS, TXN_COMPONENTS where txn_id..." query should only handle p-type rows for clean-aborted 'compaction' or non-p-type records.

We need to have 2 clean() methods now - one the existing one, the other for clean-aborted operation. The new one, should batch as much work in a single go as possible, i.e. find all TXN_COMPONENTS entries with p-type and delete all relevant deltas in 1 scan. Since it only deals with aborted deltas, it doesn't need to set the "minOpenTxnId" or use it at all.

Since doing “ls” over large number of objects may be expensive, we should introduce a ThreadPool in Cleaner to run actual clean() operations. Perhaps this pool should use a priority queue to prioritize clean-aborted items.

Hive 3.0 Upgrade Process

The upgrade works by taking a snapshot of HDFS/HMS DB and then installing 3.0 code. If something goes wrong, the ‘rollback’ is to throw away the new 3.0 cluster and restore everything from backups for the 2.x cluster. So an old version of Hive is not expected to be able to read newer HMS DB schema/data.