Problem Statement

For a given pair of Data centers DC-1 / DC-2 they can have active policies simultaneously running across both DC's on disjoint DB's. Currently repl dump aborts all open transactions before doing a bootstrap dump. Given this, if we abort any replication related transaction on a different DB to do a bootstrap dump, it will lead to an inconsistent state.

Solution

We can't take a bootstrap dump if there are open transactions. If there are open transactions during the bootstrap phase, we won't be able to replicate the data during incremental catch up and will lead to data loss. But the question is do we need to abort all open transactions? We have categorised the type of open transactions and based on that check if they really need to be aborted.

- **Read transactions** : They needn't be aborted. Read transactions are of type READ_ONLY. So we can classify them and not abort transactions of type READ_ONLY. This is already handled in the get_open_transactions in HMS. It doesn't return READ_ONLY transactions.
- **Write transactions** : They need to be aborted else will lead to data loss. We need to get the write transactions for the db under replication. We can get that from the HIVE_LOCKS table. It has a mapping for transaction id and db related info. We get all the lock info for the db under replication from HIVE_LOCKS table. The lock info will contain the transaction id as well. Check in the list of open transactions if any of them belong to this database. We need to wait for the write transaction to be completed for the configured time. If the transaction is still open, abort the transaction and go ahead with the repl dump. The other way is to expose another configuration which allows you to abort the open transactions. If disabled, we abort the bootstrap repl dump if there are open write transactions even after the specified time configuration.
- **Repl related transactions** : The repl related transactions will definitely not belong to this database as repl load and dump won't be done on the same database. The repl transaction is special and of transaction type REPL_CREATED. The transaction type REPL_CREATED was initially introduced in order to avoid aborting the repl transactions even if there is no heartbeat. We can reuse the same for our use case. It has a field for repl policy which contains the database name also. So we can easily classify repl transactions and not abort them if a bootstrap dump is in progress.

Final Approach

1. Get_open_transactions doesn't return READ_ONLY transactions. So read transactions are not getting aborted already. Nothing needs to be done for them.

2. Classify repl related transactions and don't abort them. Transaction type(REPL_CREATED) will help in that.
3. Expose a config to enable aborting write transactions.
4. If enabled, abort the write transactions for the db under replication after waiting for the time configured by **hive.repl.bootstrap.dump.open.txn.timeout.** If disabled, fail the current bootstrap dump and try again in the next schedule.
5. To get the write transactions only for the db under replication we do the following : We can get that from the HIVE_LOCKS table. It has a mapping for transaction id and db related info. We get all the lock info for the db under replication from HIVE_LOCKS table. The lock info will contain the transaction id as well. Check in the list of open transactions if any of them belong to this database.
6. Checking the HIVE_LOCKS table to get the write transactions for the db will be done after the configured time(**hive.repl.bootstrap.dump.open.txn.timeout**) just before we abort the transactions. This is done to simplify the waiting logic. We expect locks to be taken for any open transactions at least till this configured time.
7. **As a future optimization**, checking the HIVE_LOCKS can be done before waiting for the entire wait time period. We need to take into account the time taken to acquire the lock after the transaction started.