enis@apache.org, jmhsieh@apache.org,

3/1/13

HBASE-7305 ZK RW TABLE LOCKS

Goal

- Prevent Multiple Table operations from interfering with each other.
  - Ex: Do not allow delete while enabling table
  - Ex: Do not allow snapshot while deleting a table, etc.

Rules (intuitively)

- Like a RWLock
  - Read lock for…
  - Write lock for…
- Is the W lock reentrant?

Where it is used Usage

- EventHandler.prepare
  - Needs to be called before EventHandler.process

- Take TableWriteLock
  - CreateTableHandler
  - DisableTableHandler
  - EnableTableHandler
  - TableEventHandler
    - TableModifyFamilyHandler
    - TableAddFamilyHandler
    - TableDeleteFamilyHandler
    - ModifyTableHandler
    - DeleteTableHandler
- Splitting?
- Hbck?
- Balancer?

Config and Externally Visible ZK Stuff

- Conf hbase.table.lock.enable
- Conf zookeeper.znode.tableLock
  - Default = table-lock
  - Locks in ZK: /hbase/table-lock
  - /hbase/table-lock/<tableName>
    - Lock nodes are…
      - /hbase/table-lock/<tableName>/read-xxx
      - /hbase/table-lock/<tableName>/write-xxx
    - Node contains metadata as Ephemeral-sequential node in table-lock/<tablename> dir? (Why ephemeral?)

Table Lock Metadata Protobuf

- Bytes tableName
- ServerName lockOwner
- Int64 threadId
- Bool isShared
- String purpose

Classes

- Interface InterProcessLock
  - ZKInterProcessLockBase
    - ZKInterProcessReadLock
    - ZKInterProcessWriteLock
- Interface InterProcessReadWriteLock
  - ZKInterprocessReadWriteLock

- TableLockManager / interface TableLock
  - TableLockManager.ZKTableLockManager
  - In HMaster wit ref in Assignment Manager

TableLockManager Life Cycle

- Upon Create, reap at master side
  - (only if not in master recovery mode)
- Base:
  - Watch parentLock node
- Action:
  - TableLockManager.writeLock
  - TableLockManager.readLock