

HBase Consensus

Mikhail Antonov [~mantonov]

April 14, 2014

See <https://issues.apache.org/jira/browse/HBASE-10909> for reference

Motivation

In <https://issues.apache.org/jira/browse/HBASE-10296> the discussion has started about how to replace ZK by consensus library to better handle HMaster failover and state consistency.

Whatever algorithm / implementation may be chosen (let's assume we have a library implementing such an algorithm), some refactoring of the current HBase codebase is needed to integrate it. Now low-level ZK-related API (ZkUtil, ZooKeeperWatcher classes for example) is used throughout the codebase. So one of the first practical steps seems to be to abstract out ZK API by identifying logical constructs implemented now via ZK and providing pluggable API for that.

What ZooKeeper is used for

At higher level, there are 3 usecases for ZooKeeper as used now in HBase:

- **Access to transient shared state**
Some information is kept in ZK and all nodes read / write it when they need (for example, list of region servers currently registered with master, or active master IP).
- **Sending notifications which other nodes need to learn about**
Creating / modifying certain znodes to trigger ZK watches on all nodes watching
- **Receiving notifications generated by other nodes**
ZooKeeperWatcher is used as a broker which receives watch-events from ZK ensemble and pushes them to all registered ZooKeeperListener's.

("other nodes" may mean different part of the cluster which may be running on the same host or within the same JVM)

Consensus model and API

3 use cases above could be abstracted out from ZK as follows (here I'm giving generalized model and how it can be implemented by ZAB/ZooKeeper and Paxos, as possible example consensus providers):

Shared state

Whenever we read / write shared state, we make a call to Consensus class (e.g. RegionServerConsensus interface):

- ZK-based implementation would do something like `ZkUtil.listChildrenNoWatch()`, or `ZkUtil.getDataNoWatch()`, which is what it does now
- Paxos-based impl would access locally-kept data (part of the application state, as defined by replicated state machine)

Sending notifications

Whenever we send notification, we make a call to appropriate Consensus class (e.g. SplitLogWorkerConsensus):

- ZK-based impl would modify certain znodes in ZooKeeper namespace to trigger subscribed watchers
- Paxos-based impl would generate a proposal to be agreed upon by acceptors quorum

Receiving notifications

Whenever certain logic is to be executed in response to certain event, we let it to be called from the Consensus class.

- ZK-based implementation would make Consensus class a subclass of `ZooKeeperListener`, registered with `ZooKeeperWatcher`, with appropriate logic for events like `nodeCreated()`, `nodeDeleted()`, `nodeDataChanded()` etc.
- Paxos-based impl may be a learner, invoking appropriate handlers when the certain proposal gets agreed upon by acceptors and learned by the learner.

Designing Consensus API

- **Redesign ZooKeeperListeners to abstract them from ZooKeeper API**

Current design:

- *ZooKeeperListener* interface is implemented by a number of classes (~25) and has 4 low-level API methods - *nodeCreated()*, *nodeDeleted()*, *nodeDataChanged()*, *nodeChildrenChanged()*
- *ZooKeeperWatcher* maintains list of listeners and upon receiving ZK event calls all registered listeners and passes them affected znode's path.

Proposed design:

- Create new interface *ConsensusHandler*, with single method *handle(Agreement agreement)*
- Modify classes currently implementing *ZooKeeperListener* interface so instead they implement new *ConsensusHandler* interface, and in their *handle(...)* methods they will typecast the *agreement* instance to specific type (like *AssignmentAgreement*), extract information encoded in it, and do their work
- Create interface called *AgreementLearner* with implementation *ZkAgreementLearner*, which will maintain a list of all *ConsensusHandlers* and using znode prefixes (used for events dispatching, like *assignmentZNode* or *balancerZNode*, and kept in *ZooKeeperWatcher*) can parse znodes-level event, construct Agreement objects of proper type and dispatch them to appropriate *ConsensusHandler's*
- Modify *ZooKeeperWatcher*:
 - remove internal collection of *ZooKeeperListeners*
 - in *handle(...)* method pass the event to *ZkAgreementLearner* for further dispatching and execution.

Alternative implementation of *AgreementLearner* would (instead of getting called from *ZooKeeperWatcher*) be called by Paxos or Raft listener.