Fsck and autorecovery

BookKeeper auto recovery discussed in BOOKKEEPER-237 JIRA and already implemented many sub-tasks in it.
We have to discuss about Fsck feature. Edit this page

State of the plan (as of 31 Aug 2012)

BookKeeper Auto-Recovery

When any Bookie goes down in the BookKeeper cluster, there is no way to recover the lost data from that Bookie server. For example, if we have 2 replicas for a ledger in BK cluster, and a node goes down from it, we will be running the cluster with single replica. Running clusters with single or no replicas will be a risk, as nodes may fail in general. To avoid such situations, we need a mechanism for recovering the data to new bookies for meeting the enough replica criteria (quorum size) and it is called Auto-Recovery in BookKeeper.

Working

Auto-Recovery has two main modules:

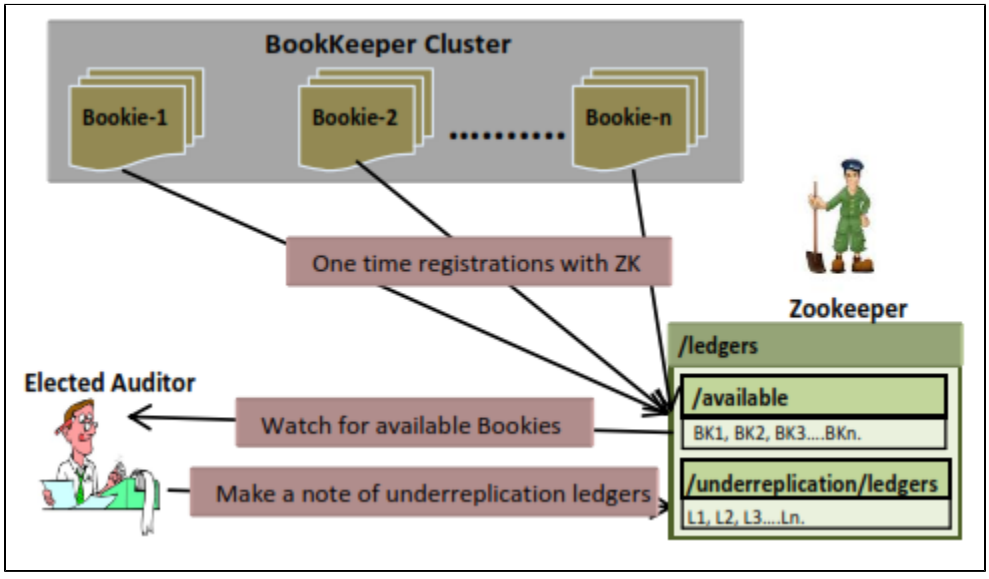
- **Auditor**
- **ReplicationWorker**

AutoRecoveryMain is an Auto-recovery node, which internally initializes and starts Auditor and ReplicationWorker threads. So, each Auto-recovery node will have two threads running.

This Auto-recovery node has to be started in each Bookie machine. All recovery nodes will participate in leader election and one Auditor may become the leader and others will just watch the elected auditor failure to participate again in next election.

Auditor:

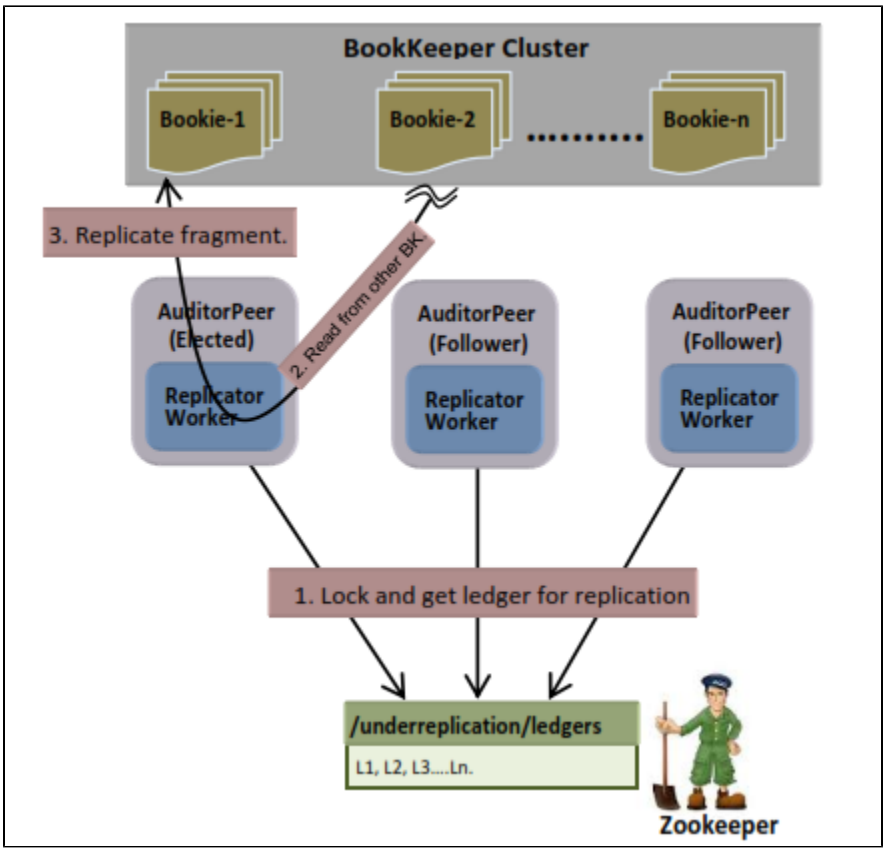
Once the Auditor thread is started, the auditor elector will go for the election to win the auditing job for Bookie cluster. Here, auditing job would be that, it has to detect the under-replicated ledgers in the cluster due to Bookie failures.



Auditor will keep watch on the available Bookies in the cluster. Bookie will add its entry in the available bookie during Bookie server startup. So, when the Bookie is crashed or killed, Auditor will get a notification about the children change in available Bookie list. Auditor will immediately scan the complete ledger list related to that failed Bookie. On getting the details of ledgers, Auditor node publishes the under-replicated ledgers in under-replication znode path in Zookeeper. After this, again it will be watching Bookie failures (by resetting Zookeeper watcher) on available Bookie list.

ReplicationWorker:

ReplicationWorker will get the under replicated ledgers one after the other for rereplication. If there are no ledgers in under-replicated state, worker will wait for the ledgers to be added into under-replicated list in ZK by Auditor.



Once the auditor publishes about the under replicated ledgers in ZK, ReplicationWorker picks one ledger and get a lock on it. Here it will get the lock by just adding an ephemeral znode with ledger name in underreplication/locks folder. Then replication worker scans the ledger and gets the under replicated fragments. If the ledger fragment already contains the local Bookie address in its ensemble, it will skip the replication for that ledger as ReplicationWorker will treat local Bookies as a target bookie to copy or replicate ledger fragment to it. If replication is not completed for all the under-replicated fragments in the ledger, ReplicationWorker will just release the lock for the ledger. Technically, releasing lock will be deleting the held lock in /underreplication/locks folder. So, that other replication worker can immediately get notified about the lock deletion and may pick this ledger for replicating the pending fragments from the ledger. This process works well for the closed ledgers. If the ledger is in the open state and if the last fragment of the ledger in under replication state, there will be a risk of data loss if we replicate the last fragment
straight away like above process.

How ReplicationWorker handled this data loss scenario?
Scenario: The last fragment of the ledger is in under replicated state; replication worker replicates it and updates the ledger metadata with local Bookies address. Immediately, the failed Bookie started and running. Now the client resumed for adding some more entries, and it can continue with adding entries with the old Bookie. But ReplicationWorker already change the metadata for that fragment with local Bookie. That means, that client unnecessarily adding the entries to the old bookie whose address is already removed from fragment ensemble. So, this can create data loss if other bookie goes down and even though old Bookie is running fine.
To prevent this situation, ReplicationWorker will postpone the replications if the last fragment of the ledger is in open state. In such case it will just schedule a timer task for that ledger for delaying replication for such ledgers. That timer task scheduling period is configurable and default value is 30000ms. Once the timer fired, it will force fence the ledger if it is still in open state and will release the ledger lock.So, that will trigger rereplication automatically as RW will loop to get the under replicated ledgers. So, any under-replicated last fragment ledger will not be kept open for long time if the client is idle and not reforming ensemble for long (more than pending replication timeout.)

Open Questions

- We should also periodically check ledgers are available. Where should this run from?