MEP 30 -- Support Coordinators Primary-backup Mechanism

Current state: Under Discussion
Keywords: HA, primary-backup, RootCoord, DataCoord, QueryCoord, IndexCoord
Released:

Summary

To achieve HA by supporting primary-backup mechanism for all Coordinators(RootCoord, DataCoord, QueryCoord, IndexCoord).

Motivation

Currently, each Milvus coordinator(coord) is a single node, so there are risks of single points of failure(SPOF). As a distributed system, Milvus needs to mitigate the impact of SPOFs and provides high available service to users. This project aims to support basic primary-backup mechanism, which is a popular high availability solution.

Public Interfaces

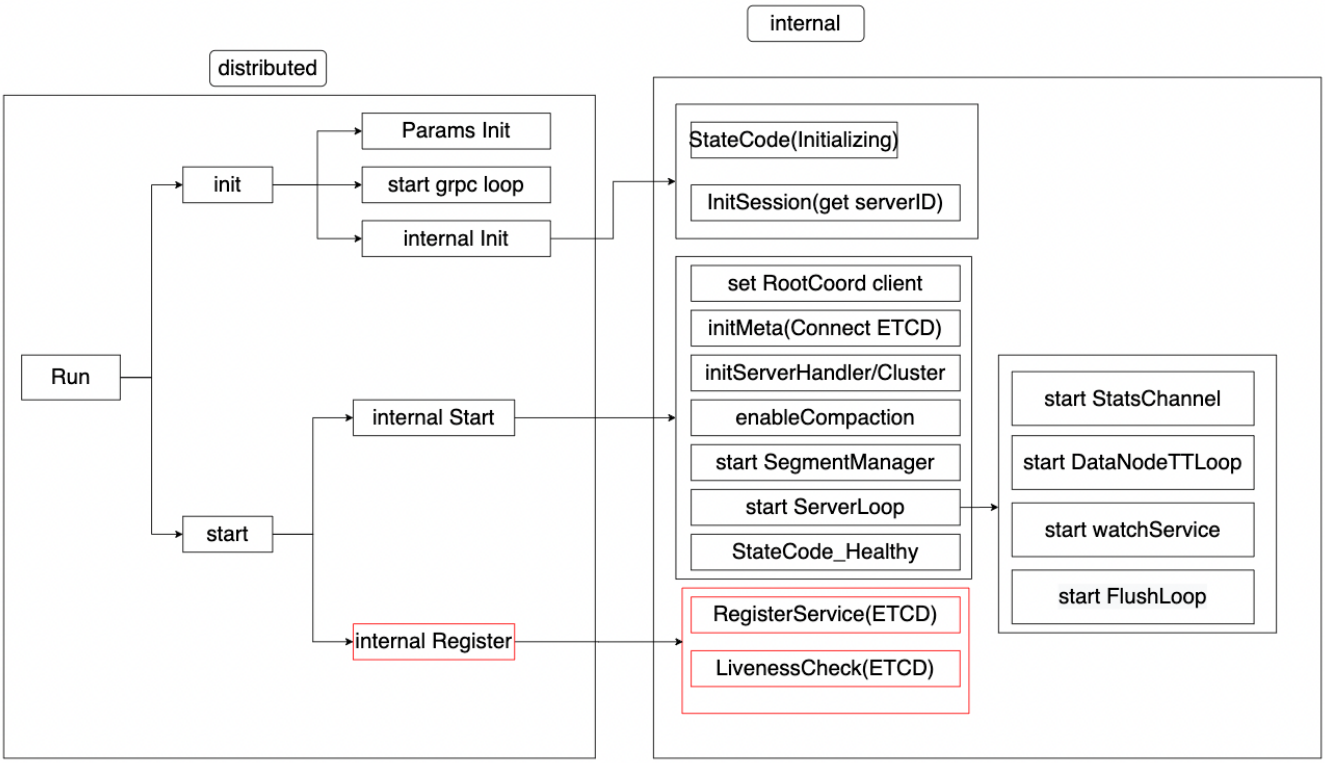
No public interfaces need to be added or changed. It's transparent to users.

Design Details

Key points for primary-backup mechanism

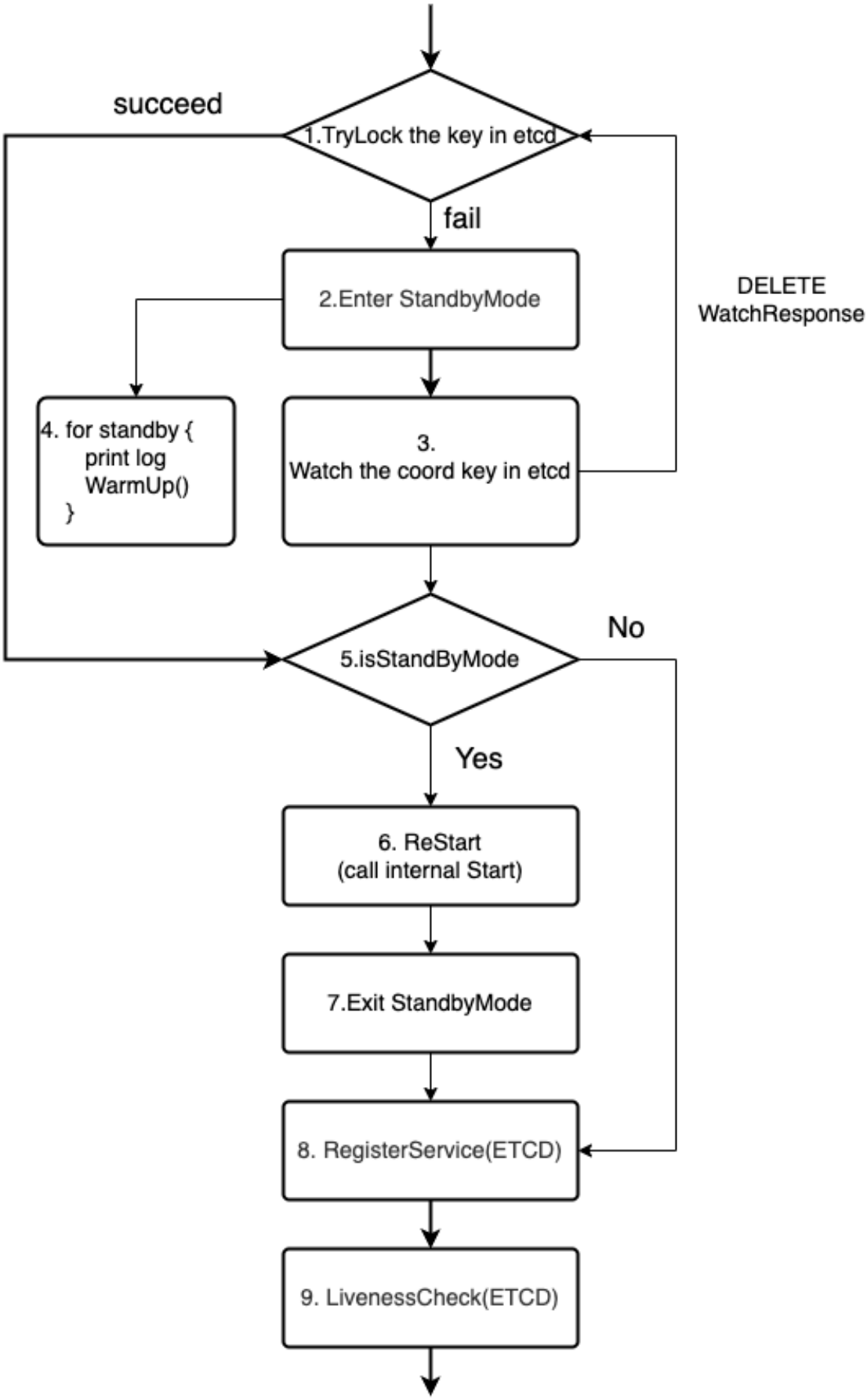
1. Failure Detection: detect primary failure quickly and accurately
2. Take Over & FailOver:
    a. The standby coord should take over the responsibility after detecting the primary coord's failure.
    b. All clients connected to that coord should failover and update the new coord info in memory automatically. (seems this is already supported)
3. Recovery: The old master can serve as a standby after recovery.

Current start-up process



The start-up progress of a coordinator is like the graph above (take DataCoord as an example, others are very similar). Currently, each component in Milvus cluster maintains a keepAlive lease and register the node info in ETCD (internal Register in the graph). Milvus establishes service discovery by querying the registered info. If a node crash, the lease will die and the related registered info will vanish. Therefore, we can easily detect failure by watching the coords' key in ETCD. The change will mainly be in internal register as followed.

internal Register


succeed
1.TryLock the key in etcd
fail
2.Enter StandbyMode
DELETE
WatchResponse
4. for standby {
print log
WarmUp()
}
3.
Watch the coord key in etcd
5.isStandByMode
No
Yes
6. ReStart
(call internal Start)
7.Exit StandbyMode
8. RegisterService(ETCD)
9. LivenessCheck(ETCD)

1. Try lock the key in etcd. If succeed, this node will become the primary. 5. If fail which means there is another node hold the lock. 2.
2. Enter StandyMode. 3, 4
3. Watch the primary key in ETCD. When receiving a key DELETE response, 1, campaign the lock.
4. Start a loop goroutine to print log and do WarmUp(). WarmUp is to do something like update the meta to accurate the Restart if necessary.
5. If it is in StandyMode. If true 6. If false 8.
6. Restart: call internal Start. 7
7. Exit StandbyMode. It will stop the loop in 4. 8
8. Register the service to ETCD as primary. 9
9. Start up the LivenessCheck goroutine

Test Plan

1, Deploy a cluster with primary and standby coords. Manually remove the primary coord or mock some crash in the primary coord. The standby coord should take over successfully. The cluster must keep working after a short time of partial failure.

2, Deploy multiple backup coordinators. There should be only one of them can turned into active.

3, The test should be done for each kind of coords.

Future work