Kafka Controller Internals

In a Kafka cluster, one of the brokers serves as the controller, which is responsible for managing the states of partitions and replicas and for performing administrative tasks like reassigning partitions. The following describes the states of partitions and replicas, and the kind of operations going through the controller.

PartitionStateChange:

Valid states are:

- NonExistentPartition: This state indicates that the partition was either never created or was created and then deleted.
- NewPartition: After creation, the partition is in the NewPartition state. In this state, the partition should have replicas assigned to it, but no leader /isr yet.
- OnlinePartition: Once a leader is elected for a partition, it is in the OnlinePartition state.
- OfflinePartition: If, after successful leader election, the leader for partition dies, then the partition moves to the OfflinePartition state.

Valid state transitions are:
NonExistentPartition -> NewPartition

1. load assigned replicas from ZK to controller cache

NewPartition -> OnlinePartition

1. assign first live replica as the leader and all live replicas as the isr; write leader and isr to ZK
2. for this partition, send LeaderAndIsr request to every live replica and UpdateMetadata request to every live broker

OnlinePartition,OfflinePartition -> OnlinePartition

1. select new leader and isr for this partition and a set of replicas to receive the LeaderAndIsr request, and write leader and isr to ZK
    a. OfflinePartitionLeaderSelector: new leader = a live replica (preferably in isr); new isr = live isr if not empty or just the new leader if otherwise; receiving replicas = live assigned replicas
    b. ReassignedPartitionLeaderSelector: new leader = a live reassigned replica; new isr = current isr; receiving replicas = reassigned replicas
    c. PreferredReplicaPartitionLeaderSelector: new leader = first assigned replica (if in isr); new isr = current isr; receiving replicas = assigned replicas
    d. ControlledShutdownLeaderSelector: new leader = replica in isr that's not being shutdown; new isr = current isr - shutdown replica; receiving replicas = live assigned replicas
2. for this partition, send LeaderAndIsr request to every receiving replica and UpdateMetadata request to every live broker

NewPartition,OnlinePartition -> OfflinePartition

1. nothing other than marking partition state as Offline

OfflinePartition -> NonExistentPartition

1. nothing other than marking the partition state as NonExistentPartition

ReplicaStateChange:

Valid states are:

1. NewReplica: When replicas are created during topic creation or partition reassignment. In this state, a replica can only get become follower state change request.
2. OnlineReplica: Once a replica is started and part of the assigned replicas for its partition, it is in this state. In this state, it can get either become leader or become follower state change requests.
3. OfflineReplica : If a replica dies, it moves to this state. This happens when the broker hosting the replica is down.
4. NonExistentReplica: If a replica is deleted, it is moved to this state.

Valid state transitions are:

NonExistentReplica --> NewReplica

1. send LeaderAndIsr request with current leader and isr to the new replica and UpdateMetadata request for the partition to every live broker

NewReplica-> OnlineReplica

1. add the new replica to the assigned replica list if needed

OnlineReplica,OfflineReplica -> OnlineReplica

1. send LeaderAndIsr request with current leader and isr to the new replica and UpdateMetadata request for the partition to every live broker

NewReplica,OnlineReplica -> OfflineReplica

1. send StopReplicaRequest to the replica (w/o deletion)

2. remove this replica from the isr and send LeaderAndIsr request (with new isr) to the leader replica and UpdateMetadata request for the partition to every live broker.

OfflineReplica -> NonExistentReplica

1. send StopReplicaRequest to the replica (with deletion)

KafkaController Operations:

onNewTopicCreation:

1. call onNewPartitionCreation

onNewPartitionCreation:

1. new partitions -> NewPartition
2. all replicas of new partitions -> NewReplica
3. new partitions -> OnlinePartition
4. all replicas of new partitions -> OnlineReplica

onBrokerFailure:

1. partitions w/o leader -> OfflinePartition
2. partitions in OfflinePartition and NewPartition -> OnlinePartition (with OfflinePartitionLeaderSelector)
3. each replica on the failed broker -> OfflineReplica

onBrokerStartup:

1. send UpdateMetadata requests for all partitions to newly started brokers
2. replicas on the newly started broker -> OnlineReplica
3. partitions in OfflinePartition and NewPartition -> OnlinePartition (with OfflinePartitionLeaderSelector)
4. for partitions with replicas on newly started brokers, call onPartitionReassignment to complete any outstanding partition reassignment

onPartitionReassignment: (OAR: old assigned replicas; RAR: new re-assigned replicas when reassignment completes)

1. update assigned replica list with OAR + RAR replicas
2. send LeaderAndIsr request to every replica in OAR + RAR (with AR as OAR + RAR)
3. replicas in RAR - OAR -> NewReplica
4. wait until replicas in RAR join isr
5. replicas in RAR -> OnlineReplica
6. set AR to RAR in memory
7. send LeaderAndIsr request with a potential new leader (if current leader not in RAR) and a new assigned replica list (using RAR) and same isr to every broker in RAR
8. replicas in OAR - RAR -> Offline (force those replicas out of isr)
9. replicas in OAR - RAR -> NonExistentReplica (force those replicas to be deleted)
10. update assigned replica list to RAR in ZK
11. update the /admin/reassign_partitions path in ZK to remove this partition
12. after electing leader, the replicas and isr information changes, so resend the update metadata request to every broker

For example, if OAR = {1, 2, 3} and RAR = {4,5,6}, the values in the assigned replica (AR) and leader/isr path in ZK may go through the following transition.

AR	leader/isr	
{1,2,3}	1/{1,2,3}	(initial state)
{1,2,3,4,5,6}	1/{1,2,3}	(step 2)
{1,2,3,4,5,6}	1/{1,2,3,4,5,6}	(step 4)
{1,2,3,4,5,6}	4/{1,2,3,4,5,6}	(step 7)
{1,2,3,4,5,6}	4/{4,5,6}	(step 8)
{4,5,6}	4/{4,5,6}	(step 10)

Note that we have to update AR in ZK with RAR last since it's the only place where we store the OAR persistently. This way, if the controller crashes before that step, we can still recover.

onControllerFailover:

1. replicaStateMachine.startup():
    a. initialize each replica to either OfflineReplica or OnlineReplica
    b. each replica -> OnlineReplica (force LeaderAndIsr request to be sent to every replica)
2. partitionStateMachine.startup():
    a. initialize each partition to either NewPartition, OfflinePartition or OnlinePartition
    b. each OfflinePartition and NewPartition -> OnlinePartition (force leader election)
3. resume partition reassignment, if any
4. resume preferred leader election, if any

onPreferredReplicaElection:

1. affected partitions -> OnlinePartition (with PreferredReplicaPartitionLeaderSelector)

shutdownBroker:

1. each partition whose leader is on shutdown broker -> OnlinePartition (ControlledShutdownPartitionLeaderSelector)
2. each replica on shutdown broker that is follower, send StopReplica request (w/o deletion)
3. each replica on shutdown broker that is follower -> OfflineReplica (to force shutdown replica out of the isr)