KIP-236: Interruptible Partition Reassignment

Authors: George Li, Tom Bentley

Status

Current state: Under Discussion

Discussion thread: *here*

JIRA: *KAFKA-6359*

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

The current `kafka-reassign-partitons.sh` tool imposes the limitation that only a single batch of partition reassignments can be in-flight, and it is not possible to cancel a reassignment that is in-flight cleanly, safely in a timely fashion (e.g. as reported KAFKA-6304, the current way of reassignment cancellation requires a lot of manual steps). This has a number of consequences:

1. Reassignments especially for large topic/partition is costly. In some case, the performance of the Kafka cluster can be severely impacted when reassignments are kicked off. There should be a fast, clean, safe way to cancel and rollback the pending reassignments. e.g. original replicas `[1,2,3]`, new replicas `[4,5,6]`, causing performance impact on Leader 1, the reassignment should be able to get cancelled immediately and reverted back to original replicas `[1,2,3]`, and dropping the new replicas.
2. Each batch of reassignments takes as long as the slowest partition; this slowest partition prevents other reassignments from happening. This can be happening even in the case submitting the reassignments by grouping similar size topic/partitions into each batch. How to optimally group reassignments into one batch for faster execution and less impact to the cluster is beyond the discussion in this KIP. This is addressed in the Planned Future Changes section and may be implemented in another KIP.
3. Currently, the reassignment operations are still communicated directly with the Zookeeper. Other admin types of operation like create/delete topics, etc. are moving to the RPC based KIP-4 wire protocol. By moving from interacting directly with Zookeeper to RPC, it offers the user the recommended path and discourages directly modifying the Zookeeper nodes. This will pave the way to lock down Zookeeper security by ACLs, that only brokers need to communicate with ZK.

This change would enable

- Cancel all pending reassignments currently in `/admin/reassign_partitions` and revert them back to their original replicas.
- Development of an AdminClient API which supported the above features. Change the current administrative APIs to go through RPC instead of Zookeeper.

Public Interfaces

Strictly speaking this is not a change that would affect any public interfaces (since ZooKeeper is not considered a public interface, and it can be made in a backward compatible way), however since some users are known to operate on the `/admin/reassign_partitions` znode directly, this could break in future versions of Kafka (e.g. as reported in KAFKA-7854), and such operations should be discouraged.

A new znode `/admin/cancel_reassignment_in_progress` is used to signal the Controller to cancel current pending reassignments in `/admin/reassign_partitions`, Note that we can only cancel the pending reassignments of current batch of reassignments, some reassignments can complete almost instantly if the replicas set is not changed (already in ISR), only the ordering is changed. e.g. `(1,2,3) => (2,3,1)`, the preferred leadership is changed. To rollback all the reassignments in current batch (not just the pending reassignments, including those already completed in the same batch), the client who submitted the reassignment should keep a "rollback" version and submit as reassignment after `/admin/reassign_partitions` is empty and deleted.

For the user client submitting new reassignment JSON file format, the public interface will remain the same. The user client will submit list of topic /partition replicas (new replicas assignments). Before writing to the znode `/admin/reassign_partitions`, the controller will be adding `"original_replicas"` to support rollback to its original state of the topic partition assigned replicas. How `"original_replicas"` gets populated will be discussed in detail later.

Proposed Changes

Reassignment Cancellation

The main idea is support clean, safe cancellation of pending reassignments in `/admin/reassign_partitions` znode in a timely fashion, and support more reassignments while currently some reassignments are in-flight.

When client are submitting reassignments, it only needs to submit "replicas" (new replicas assignment) of the topic / partition. Before writing to `/admin/reassign_partitions`, the current assigned replicas (original replicas) are read from Zookeeper and added the `"original_replicas"` for that topic/partition reassignments . This `"original_replicas"` will be used for rollback of the topic/partition replicas assignment during cancellation.

e.g. after the controller reads the reassignment JSON submitting by the AdminClient, the following will be written to `/admin/reassign_partitions`:

```
{"version":1,
 "partitions":[{"topic": "foo1",
                "partition": 0,
                        "replicas": [1,2,4],
                "original_replicas": [1,2,3]
               },
               {"topic": "foo2",
                "partition": 1,
                        "replicas": [7,9,8],
                "original_replicas": [5,6,8]
               }]
}
```

For `ControllerContext.partitionBeingReassigned`, also add the `originalReplicas` to the `ReassignedPartitionsContext` class besides `newReplicas` :

```
case class ReassignedPartitionsContext(var newReplicas: Seq[Int] = Seq.empty,
                                       var originalReplicas: Seq[Int]= Seq.empty,
                                       val reassignIsrChangeHandler: PartitionReassignmentIsrChangeHandler) {
```

To trigger the reassignment cancellation, a new znode `/admin/cancel_reassignment_in_progress` is created, the controller will be informed of the reassignment cancellation via a ZooKeeper watch on this. The controller will read the current pending reassignments in `/admin/reassign_partitions` and re-populate `ControllerContext.partitionsBeingReassigned`. For each pending topic/partition reassignments, the cancellation /rollback works like below, it's like the opposite of doing reassignments, since we have the `"original_replicas"` of each topic /partition reassignments in `/admin/reassign_partitions` & `ControllerContext.partitionBeingReassigned`, it is much easier to rollback.

```
RAR = Reassigned replicas
OAR = Original list of replicas for partition
AR = current assigned replicas

1. Set AR to OAR in memory.
2. If the leader is not in OAR, elect a new leader from OAR. If new leader needs to be elected from OAR, a
LeaderAndIsr
 will be sent. If not, then leader epoch will be incremented in zookeeper and a LeaderAndIsr request will be sent.
 In any case, the LeaderAndIsr request will have AR = OAR. This will prevent the leader from adding any replica in
 OAR - RAR back in the isr.
3. Move all replicas in RAR - OAR to OfflineReplica state. As part of OfflineReplica state change, we shrink the
 isr to remove RAR - OAR in zookeeper and send a LeaderAndIsr ONLY to the Leader to notify it of the shrunk isr.
 After that, we send a StopReplica (delete = false) to the replicas in RAR - OAR.
4. Move all replicas in RAR - OAR to NonExistentReplica state. This will send a StopReplica (delete = true) to
 the replicas in RAR - OAR to physically delete the replicas on disk.
5. Update AR in ZK with OAR.
6. Update the /admin/reassign_partitions path in ZK to remove this partition.
7. After electing leader, the replicas and isr information changes. So resend the update metadata request to
every broker.
```

The proposed new option: `--cancel of` AdminClient CLI will be added to submit reassignment cancellation.

```
$ zkcli -h kafka-zk-host1 ls /kafka-cluster/admin/
[u'reassign_partitions',
 u'delete_topics']

# Current pending reassignment(s)
$ zkcli -h kafka-zk-host1 get /kafka-cluster/admin/reassign_partitions
('{"version":1,"partitions":[{"topic":"test_topic","partition":25,"replicas":[1,2,4],"original_replicas":
[1,2,3]}]}', ZnodeStat(czxid=17180484637, mzxid=17180484641, ctime=1549498790668, mtime=1549498790680,
version=1, cversion=0, aversion=0, ephemeralOwner=0, dataLength=148, numChildren=0, pzxid=17180484637))

# Cancel the pending reassignments.  and remove the throttle as well.
$ /usr/lib/kafka/bin/kafka-reassign-partitions.sh  --zookeeper kafka-zk-host1/kafka-cluster --cancel
Rolling back the current pending reassignments Map(test_topic-25 -> Map(replicas -> Buffer(1, 2, 4),
original_replicas -> Buffer(1, 2, 3)))
Successfully submitted cancellation of reassignments.
The cancelled pending reassignments throttle was removed.
Please run --verify to have the previous reassignments (not just the cancelled reassignments in progress)
throttle removed.

# This is just for illustration purpose.  In reality, the cancellation of reassignments should be pretty quick.
# The below listing of /admin might not even show cancel_reassignment_in_progress & reassign_partitions
$ zkcli -h kafka-zk-host1 ls /kafka-cluster/admin/
[u'cancel_reassignment_in_progress',
 u'reassign_partitions',
 u'delete_topics']

# After reassignment cancellation is complete.  The ZK node  /admin/cancel_reassignment_in_progress  & /admin
/reassign_partitions are gone.
$ zkcli -h kafka-zk-host1 ls /kafka-cluster/admin/
[u'delete_topics']
```

If the pending reassignments have throttle, the throttle will be removed after the reassignments are cancelled. However for the reassignments already completed, the user would need to remove their throttle by running the `kafka-reassign-partitions.sh --verify`

Skip Reassignment Cancellation Scenarios

There are a couple scenarios that the Pending reassignments in `/admin/reassign_partitions` can not be cancelled / rollback.

1. If the "original_replicas" is missing for the topic/partition in `/admin/reassign_partitions` . In this case, the pending reassignment cancelled will be skipped. Because there is no way to reset to the original replicas. The reasons this can happened could be:
    a. if either the user/client is tampering `/admin/reassign_partitions` directly, and does not have the "original_replicas" for the topic
    b. if the user/client is using incorrect versions of the admin client to submit for reassignments. The Kafka software should be upgraded not just for all the brokers in the cluster. but also on the host that is used to submit reassignments.

2. If all the "original_replicas" brokers are not in ISR, and some brokers in the "new_replicas" are not offline for the topic/partition in the pending reassignments. In this case, it's better to skip this topic's pending reassignment cancellation/rollback, otherwise, it will become offline. However, if all the brokers in "original_replicas" are offline AND all the brokers in "new_replicas" are also offline for this topic /partition, then the cluster is in such a bad state, the topic/partition is currently offline anyway, it will cancel/rollback this topic pending reassignments back to the "original_replicas".

Planned Future Changes

New reassignments while existing reassignments in-flight

The above Reassignment Cancellation is more straight forward. However, to submit new reassignments while there are existing reassignments are still in-flight, it needs a bit more discussions and consensus. It might be worth doing it in another KIP. So it's listed as Planned Future Changes, if consensus can be reached on this design, this feature can be delivered in this KIP as well.

In order to support submitting more reassignments while existing reassignments are still in-flight. An extra znode `/admin /reassign_partitions_queue` which has the same JSON format as `/admin/reassign_partitions`. Three more options `--generate-queue --verify-queue --execute-queue` will be added to `kafka-reassign-partitions.sh`. The controller will be informed of the queued reassignments via a ZooKeeper watch. It will get all topic/partitions from `/admin/reassign_partitions_queue` and add to `/admin /reassign_partitions`, then trigger the reassignments `onPartitionReassignment()` of the topic/partitions.

The new `/admin/reassign_partitions_queue` znode JSON format is the same as `/admin/reassign_partitions` . e.g. :

```
{"version":1,
 "partitions":[{"topic": "foo1",
                "partition": 0,
                        "replicas": [1,2,5]
               },
               {"topic": "foo2",
                "partition": 1,
                        "replicas": [7,9,10]
               }]
}
```

If `/admin/reassign_partitions_queue` znode already exists, new queued reassignments will be blocked from writing to `/admin/reassign_partitions_queue`.

In case inside the `/admin/reassign_partitions_queue`, there are topic/partitions which exist in `/admin/reassign_partitions` (pending reassignments), the conflict resolution for those duplicate topic/partitions is to first cancel / rollback the pending reassignments of those topic/partitions in `/admin/reassign_partitions`, then submit new reassignments from `/admin/reassign_partitions_queue` to `/admin/reassign_partitions`. This approach will be simpler than the algorithm proposed by Tom Bentley previously to infer the final replicas assignments for those duplicate topic/partitions. After the topic/partition is put in `/admin/reassign_partitions` & `ControllerContext.partitionBeingReassigned` to trigger the reassignment, the topic/partition will be removed from `/admin/reassign_partitions_queue`, and when `/admin/reassign_partitions_queue` is empty, the znode will be deleted.

Compatibility, Deprecation, and Migration Plan

As described above, compatibility with `/admin/reassign_partitions` is maintained, so existing software will continue working. The newly introduced znode `/admin/cancel_reassignment_in_progress` is used solely for canceling/rollback of current reassignments still pending in `/admin/reassign_partitions`.

This compatibility behavior could be dropped in some future version of Kafka, if that was desirable.

Rejected Alternatives

1. A similar protocol based on just `/admin/reassignments` without the `/admin/reassignment_requests` was initially considered, but that required a ZK watch per reassignment, which would not scale well. This proposal requires only 1 more watch than the current version of the broker.