KIP-497: Add inter-broker API to alter ISR

Master KIP

KIP-500: Replace ZooKeeper with a Self-Managed Metadata Quorum (Accepted)

Status

Current state: Adopted

Discussion thread: here

JIRA: ⚠ Unable to render Jira issues macro, execution error.

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

Motivation

Leader and ISR information is stored in the `/brokers/topics/[topic]/partitions/[partitionId]/state` znode. It can be modified by both the controller and the current leader in the following circumstances:

1. The controller creates the initial Leader and ISR on topic creation
2. The controller can shrink the ISR as part of a controlled shutdown or replica reassignment
3. The controller can elect new leaders at any time (e.g. preferred leader election)
4. Leaders can expand or shrink the ISR as followers come in and out of sync.

Since the znode can be modified by both the controller and partition leaders, care must be taken to protect updates. We use the zkVersion of the corresponding znode to protect updates, which means we need a mechanism to propagate it between the controller and leaders and vice versa. Currently, controllers propagate the zkVersion to leaders through the LeaderAndIsr request. Leaders, on the other hand, propagate the zkVersion to the controller by creating a sequential notification znode that the controller watches. Propagation from the leader to the controller is delayed (currently by one minute), which means Metadata is typically a bit slow to reflect ISR changes made by the leader.

In this KIP, we propose a new AlterIsr API to replace the notification znode in order to give the controller the exclusive ability to update Leader and ISR state. Leaders will use this API to request an ISR change from the controller rather than directly mutating the underlying state. This ensures that the controller will always have the latest Leader and ISR state for all partitions. Concretely, this has the following benefits:

- It will not be possible for the controller to send stale metadata through the LeaderAndIsr and UpdateMetadata APIs. New requests will always reflect the latest state.
- The controller can reject inconsistent leader and ISR changes. For example, if the controller sees a broker as offline, it can refuse to add it back to the ISR even though the leader still sees the follower fetching.
- When updating leader and ISR state, it won't be necessary to reinitialize current state (see KAFKA-8585). Preliminary testing shows this can cut controlled shutdown time down by as much as 40% (take this with a big grain of salt).
- Partition reassignments complete only when new replicas are added to the ISR. With this change, reassignments can complete sooner because the controller does not have to await change notification.

Below we discuss the behavior of the new API in more detail.

Public Interfaces

We will introduce a new AlterIsr API which requires CLUSTER_ACTION permission (similar to other InterBroker APIs such as LeaderAndIsr). The request and response schema definitions are provided below:

```
AlterIsrRequest => BrokerId BrokerEpoch [Topic [PartitionId LeaderAndIsr]]
  BrokerId => Int32
  BrokerEpoch => Int64
  Topic => String
  PartitionId => Int32
  LeaderAndIsr => Leader LeaderEpoch Isr CurrentZkVersion
    Leader => Int32
    LeaderEpoch => Int32
    Isr => [Int32]
    CurrentZkVersion => Int32

AlterIsrResponse => ErrorCode [Topic [PartitionId ErrorCode]]
  ErrorCode => Int16
  Topic => String
  PartitionId => Int32
```

Possible top-level errors:

- CLUSTER_AUTHORIZATION_FAILED
- STALE_BROKER_EPOCH
- NOT_CONTROLLER

Partition-level errors:

- FENCED_LEADER_EPOCH: There is a new leader for the topic partition. The leader should not retry.
- INVALID_REQUEST: The update was rejected due to some internal inconsistency (e.g. invalid replicas specified in the ISR)
- INVALID_ISR_VERSION (NEW): The (Zk)version in the request is out of date. Likely there is a pending LeaderAndIsr request which the leader has not yet received.
- UNKNOWN_TOPIC_OR_PARTITION: The topic no longer exists.

Proposed Changes

The main change in this KIP is to send the AlterIsr request to the controller when shrinking or expanding the ISR on the leader. The controller will update the state asynchronously and send a LeaderAndIsr request once the change has been completed.

The basic approach is modeled here. The leader will not block while awaiting the ISR change. There are two cases to consider:

- **ISR Shrink**: it is not safe for the leader to assume a smaller ISR until the update has been successfully written. If we allow the high watermark to advance and the ISR shrink ultimately fails, then we risk violating replication semantics if the replica being evicted becomes leader. Therefore the leader will continue relying on the previous ISR until the LeaderAndIsr with the successful change has been received.
- **ISR Expand**: it is always safe to let high watermark advance according to a superset of the ISR. When expanding the ISR, the leader will assume the larger set immediately and delay high watermark advancement.

Basically the leader always assumes the largest possible ISR (taking int account inflight requests) when advancing the high watermark.

Today we get a performance benefit from batching leader and ISR updates to the controller by delaying propagation. This results in fewer LeaderAndIsr and UpdateMetadata updates. The approach here allows us to retain the same benefit since the controller can choose when to propagate updates.

Improved batching: We are also providing the opportunity for additional batching when updating Zookeeper. For example, ISR expansions following a rolling restart are often not time sensitive. In the future, the controller could delay the expansion in order to do multiple updates at once. Here we take a simpler approach: when the leader receives a Fetch request from a follower, some number of partitions will come into sync. The leader will send the AlterIsr for all of these partitions. The controller will respond as soon as it receives the request and apply the ISR updates. Once they are complete, it will send LeaderAndIsr updates for all affected partitions. This handles a common case today during a rolling restart when a broker comes back online and is added to many ISRs at once. In fact, the lack of batching for this case today introduces the risk of a replica falling out of ISR while all of these updates are being applied individually by the leader.

It can happen that a leader and ISR update fails after the AlterIsr was acknowledged. For example, if the controller fails before applying the changes, a new controller will be elected. It could also happen that there was already a pending change at the time the leader wanted to update ISR. In either case, the leader will receive a LeaderAndIsr update with the new version. The leader will check the status of current replicas at that time to see whether any additional changes are needed.

Compatibility, Deprecation, and Migration Plan

Use of the new AlterIsr API will be gated by the `inter.broker.protocol.version` config. Once the IBP has been bumped to the latest version, the controller will no longer register a watch for the leader and ISR notification znode.

Rejected Alternatives

None yet.