KIP-352: Distinguish URPs caused by reassignment

- Status
- Motivation
- Proposed Changes
- Public Interfaces
- Compatibility, Deprecation, and Migration Plan
- Rejected Alternatives

Status

Current state: Adopted

Discussion thread: here

JIRA:

⚠ Unable to render Jira issues macro, execution error.

⚠ Unable to render Jira issues macro, execution error.

⚠ 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

While a reassignment is in progress, new replicas are trying to catch up and are not in the ISR. The broker considers these partitions "under-replicated" even if the desired replication factor is always satisfied. This is misleading and makes URP metrics difficult to use for alerts. In KIP-455, we gave the leader a way to detect a reassignment. Specifically, the LeaderAndIsr request now has a separate field for the replicas which are being added and those that are being removed. This allows us to compute a more useful metric value.

Proposed Changes

We will change the semantics of the "UnderReplicated" metric to taking into account the AddingReplicas. Specifically, we will use the following formula:

```
isUnderReplicated == size(original assigned replicas) - size(isr) > 0
```

We count a partition as under-replicated if the current isr is smaller than the size of the current replica set. This allows us to count AddingReplicas which makes this metric consistent with UnderMinIsr criteria. Note that a reassignment may change the number of replicas, but URP satisfaction will not take this into account until the reassignment is complete.

Similarly, we will change the behavior of the kafka topic command so that `--under-replicated-partitions` returns results consistent with the change above. Because the adding/removing replicas are not visible from the Metadata API, we will use the new ListReassignment API.

Additionally, we are adding a couple new metrics to track the progress of an active reassignment. These are described below.

Public Interfaces

As described above, this KIP changes the semantics of `kafka.server:type=ReplicaManager,name=UnderReplicatedPartitions`. Replicas which are being added as part of a reassignment will not count toward this value.

We will also add some additional metrics to improve monitoring for reassignments. The table below shows all of the changes from this KIP.

Metric	Is New	Type	Includes Current Assigned Replicas	Includes Reassigning Replicas
kafka.server:type=ReplicaManager,name=UnderReplicatedPartitions	No	Gauge	Yes	No
kafka.server:type=ReplicaManager,name=ReassigningPartitions	Yes	Gauge	No	Yes
kafka.server:type=ReplicaManager,name=ReassignmentMaxLag	Yes	Gauge	No	Yes
kafka.server:type=BrokerTopicMetrics,name=ReassignmentBytesOutPerSec	Yes	Meter	No	Yes
kafka.server:type=BrokerTopicMetrics,name=ReassignmentBytesInPerSec	Yes	Meter	No	Yes

Note that the `` `ReassignmentBytesOutPerSec` `` and `` `ReassignmentBytesInPerSec` `` meters are broker-level metrics. We are not proposing any topic-level metrics for tracking reassignment progress.

`ReassignmentMaxLag` will be implemented separately as it requires some more consideration. JIRA is linked on the top of the KIP.

Compatibility, Deprecation, and Migration Plan

The main concern from a compatibility perspective is the semantic change to the "UnderReplicated" metric. Users may have to make changes if this is used to track the reassignment state. However, we believe that continued misuse of this metric (i.e. not taking reassignment into account) is a more substantial problem.

Rejected Alternatives

We considered leaving the "UnderReplicated" metric with its current semantics and adding a new metric to represent the "under-synchronized" replicas. We ultimately rejected this because we felt it was necessary to address the misuse of the URP metric due to its surprising behavior during a reassignment.