

KIP-542: Partition Reassignment Throttling


- [Status](#)
- [Motivation](#)
- [Public Interfaces](#)
- [Proposed Changes](#)
 - [Behavior of Existing Configs](#)
 - [New Configs](#)
 - [Reassignment Tool](#)
 - [Throttling Calculation](#)
 - [Updating Reassignment Throttling](#)
- [Compatibility, Deprecation, and Migration Plan](#)
- [Rejected Alternatives](#)

Status

Current state: Under Discussion

Discussion thread: [here](#)

JIRA:

 Unable to render Jira issues macro, execution error.

Released:

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

Motivation

[KIP-73](#) added quotas for replication but it doesn't separate normal replication traffic from reassignment. A user is able to specify the partition and the throttle rate but it will be applied to all non-ISR replication traffic. This is can be undesirable because during reassignment it also applies to non-reassignment replication and causes a replica to be throttled if it falls out of ISR. Also if leadership changes during reassignment, the throttles also have to be changed manually. KIP-455 will make brokers aware of pending reassignment and thus we'd be able to separate these two kinds of replication and we won't have to manually specify a list of replicas to throttle because the broker would be able to figure out automatically in runtime which partitions needed to be throttled based on the LeaderAndIsr request.

As part of this KIP we plan to add a broker level dynamic configs that would define throttle rates for the reassignment related replication of the leaders and the followers in bytes/seconds. As a consequence it will be possible to limit only reassignment and it won't consume too much network bandwidth.

Goals

- Reassignment traffic shouldn't have an effect on other replication or client traffic so they'll get guaranteed throughput.
- This throughput limit should be configured dynamically by the administrator so that an ongoing reassignment's throttle could be controlled on the fly.
- The quota should be able to be upgraded easily programmatically so management systems built on Kafka should be able to expose it easily.

Public Interfaces

- **Extra configs:** two extra config will be added, called `leader.reassignment.throttled.rate` and `follower.reassignment.throttled.rate`. Please see proposed changes for more info.
- **Existing configs:** `leader.replication.throttled.replicas` and `follower.replication.throttled.replicas` would keep their existing behavior and wouldn't be deprecated.
- **Reassignment tool:** the tool would use the new configs going forward instead of the old ones.

Proposed Changes

Behavior of Existing Configs

We would keep `leader.replication.throttled.replicas` and `follower.replication.throttled.replicas` because there are still scenarios where generic replication throttling is needed. One such case is a bootstrapping broker where a lot of follower partitions need to catch up with their leader but they're using up the bandwidth from the leader partitions on that broker. In this case it is still useful to set follower replication throttling so the leaders can stay in-sync. (Furthermore as a possible improvement to bootstrapping it might be useful to allow only a subset of follower partitions to replicate at a time but this falls out of the scope of this KIP.)

New Configs

Config name	Type	Default	Valid values	Importance	Dynamic update mode
<code>leader.reassignment.throttled.rate</code>	Long	-1	<code>[-1,...]</code>	medium	per-broker
<code>follower.reassignment.throttled.rate</code>	Long	-1	<code>[-1,...]</code>	medium	per-broker

These new configs would control how much reassignment traffic can take place on a broker on the leader and the follower side. They are both maximum values which means that the actual reassignment traffic can be smaller or even zero (if there's nothing to reassign). Their maximum value is the respective leader or follower.replication.throttled.rate. Specifying a higher value would result in a configuration error.

The possible configuration variations are:

- `replication.throttled.rate` is set but `reassignment.throttled.rate` isn't (or -1): any kind of replication (so including reassignment) can take up to `replication.throttled.rate` bytes.
- `replication.throttled.rate` and `reassignment.throttled.rate` both set: reassignment can use a bandwidth up to the configured limit and the total replication limit will be `reassignment.throttled.rate + replication.throttled.rate`.
- `replication.throttled.rate` is not set but `reassignment.throttled.rate` is set: in this case general replication has no bandwidth limits but reassignment.throttled.rate has the configured limits.
- neither `replication.throttled.rate` nor `reassignment.throttled.rate` are set (or -1): no throttling is set on any replication.

It is useful to add this feature on both leader and follower side as throttling only on the leader for instance make it more complicated to calculate the throughput limit on the follower side. For instance we may have 2 reassigning partitions with the overall limit of 10MB/s configured and that would mean 20MB/s used bandwidth on a broker which is replicating from those partitions. But since replicas can scale up the thousands on a single broker, the complexity of calculating the resulted follower reassignment would increase proportionally.

Behavior-wise they'd throttle the `addingReplicas` of the `LeaderAndIsrRequest` during reassignment.

To change these configs, the user must have `ALTER_CONFIG` privilege on the given cluster config as imposed by the `incrementalAlterConfigs` API which will be the medium for applying the configuration. By adding the new configs we'd also like to remove the related zookeeper dependencies in the `kafka-reassign-partitions.sh` command, so applying the quota would happen through the `AdminClient` API.

Reassignment Tool

The `--throttle` option's behavior in `kafka-reassign-partitions.sh` would change as it would use the new configs going forward. If there is need for reproducing the old behavior then it would still be possible by calling `kafka-configs.sh` manually before and after the reassignment to set the correct replication throttling.

Throttling Calculation

The quota calculation method that is introduced in KIP-73 wouldn't change in principal but we will only apply it to the calculated (reassigned) replicas accordingly. This has the benefit that we don't need to change the recommendations in KIP-73.

Updating Reassignment Throttling

At this point this KIP doesn't aim to add new `AdminClient` RPC call as the config value can be changed by the `IncrementalAlterConfigs` API.

Compatibility, Deprecation, and Migration Plan

The only change which needs to be mentioned is the tooling change. With this we'll change the `--throttle` option's behavior. If for some reason the old behavior is needed it can be reproduced by calling `kafka-configs.sh` manually before and after the reassignment with the intended parameters.

Rejected Alternatives

Reassignment throttling can be considered as a subset of replication throttle. This means that the full throttle value is given by `replication.throttle.rate` while `reassignment.throttle.rate` tells how much of that can be used for reassignment. Practically if `replication.throttle.rate` is set to 20 and `reassignment.throttle.rate` to 5, then 5 can be used for reassignment and 15 for other replication. While it has the advantage of being conceptually good in terms of handling reassignment as a special case, it is actually operationally harder to handle. If quotas are not used normally but only during reassignment as a safety net, then thinking about `replication.throttle.rate` and `reassignment.throttle.rate` together is more complicated. If `replication.throttled.rate` is configured during normal operation, then increasing or setting `reassignment.throttled.rate` would involve changing `replication.throttled.rate` to keep the bandwidth used for non-reassignment replication the same. This problem doesn't exist when these quotas are additive (meaning that 20 for replication and 5 reassignment adds up to a total of 25).

