KIP-250 Add Support for Quorum-based Producer Acknowledgment

This page is meant as a template for writing a KIP. To create a KIP choose Tools->Copy on this page and modify with your content and replace the heading with the next KIP number and a description of your issue. Replace anything in italics with your own description.

Status

Current state: *Under Discussion*

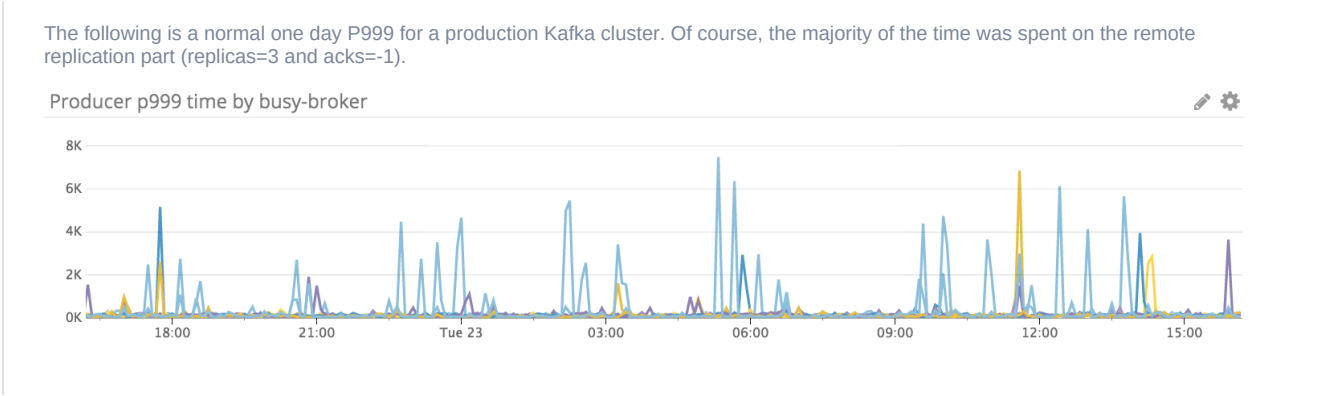
Discussion thread: *here [Change the link from the KIP proposal email archive to your own email thread]*

JIRA: *KAFKA-6477*

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

Motivation

While using "acks=-1" in the producer config, we found the P999 latency is very spiky. Since the produce request can only be acknowledged after all of the "insync.replicas" committed the message, the produce request latency is determined by the slowest replica inside "insync.replicas". In the production environment, we usually set "replica.lag.time.max.ms" to the order of 10 seconds (to protect frequent ISR shrink and expand), and producer P999 can jump to the value of "replica.lag.time.max.ms" even without any failures or retries since there can be followers lagging behind by multiple seconds and still in the "insync.replicas".



We already have pretty stable and low P99 latency, it will definitely make Kafka more suitable for a much wider audience (even in the critical write path) if we can have the similar guarantees for P999 (stable and reasonable latency).

Public Interfaces

There will be a new config on the topic level called "quorum.required.acks". The config value is an integer with default value to -1, which means this feature is turned off.

"quorum.required.acks" should be < "replicas" (if we set to "replicas", the latency can be even worse than the acks=-1) AND > 1 (if we set to 1 we should just turn this feature off and use acks=1).

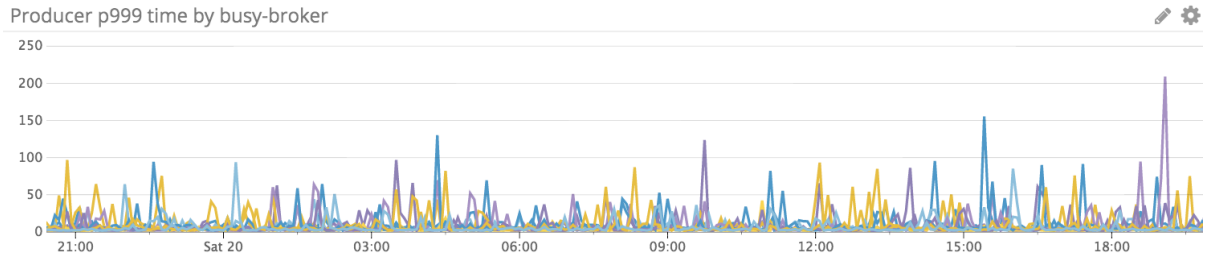
Proposed Changes

I totally understand in the early days, we have the support to specify the "request.required.acks" or "acks=2" on the producer side. We got rid of these supports since it was misleading and can't guarantee there is no data loss in all of the scenarios (and we introduced the "insync.replicas"). I am NOT against any of today's behaviors (actually, it is quite clean and straightforward), I am proposing to add an alternative to our replication semantic, which can be easily enabled/disabled, and without any impact on the existing code path.

The proposed changes here will contain two separate parts. 1: introduce a config on the topic side similar to the "request.required.acks" on the producer side previously; however, with more strict requirements. 2: improve the new leader election process to achieve the same level of data durability as of today's "insync.replicas".

1. *introduce a new config on the topic side called "quorum.required.acks". This config specifies the number of replicas a message should be committed to before the message got acknowledged. This config should be < the number of replicas and > 1. In this case, we have the data durability guarantee for "quorum.required.acks - 1" broker failures.*
2. *select the live replica with the largest LEO as the new leader during leader election. For the most popular configs in today's production environment, we use replicas=3, ack=-1, and min.insync.replicas=2. In this case, we guarantee a committed message will NOT be lost if only 1 broker failed/died. If we use replicas=3 and quorum.required.acks=2, for any message got acknowledged, it has been committed to at least 2 brokers, if the leader died, when we chose the replica with the larger LEO from the two live replicas, it will ensure this replica contains all of the messages from the previous leader. [Will document more details about how/when to get the LEO from the live replicas during the leader election]*

> The following is the P999 for the replicas=4 and quorum.required.acks=2 configs, 4000QPS with the 2KB payload for 24 hours. (I deployed with the modified version supporting quorum.required.ack based on 0.10.2.1) For the configs with replicas=3 and quorum.required.acks=2, we have very similar result.



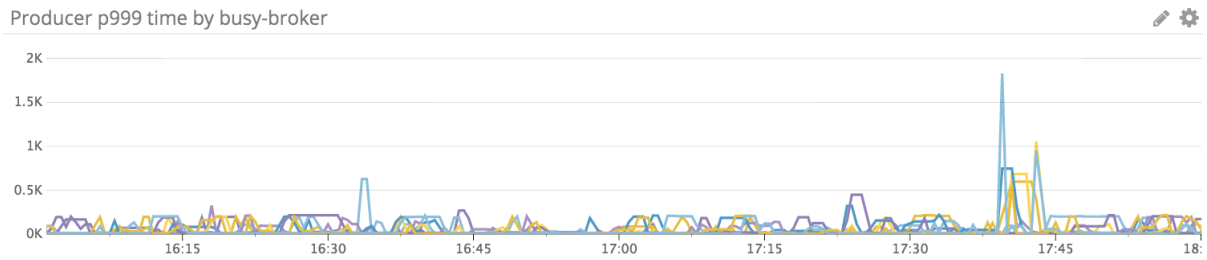
Compatibility, Deprecation, and Migration Plan

- *No impact on existing users.*
- *No need to phase out the older behavior.*
- *No migration tools required.*
- *No need to remove any existing behaviors.*

Rejected Alternatives

I got some recommendations from the offline discussions. One of them is to set replicas=2 and ack=-1, this will only wait 1 follower fetch the pending message; however, from the experiment, the P999 is still very spiky.

> The following is the P999 for the replicas=2 and ack=-1 configs, 1000QPS with the 2KB payload for only 2 hours.



Tentative PR

I posted my [PR|https://github.com/mangobatao/kafka/pull/1] in my repo, which is based on 0.10.2.1.