KIP-854 Separate configuration for producer ID expiry

Status

Current state: Accepted

Discussion thread: https://lists.apache.org/thread/cz9x90883on98k082qd0tskj6yjhox1t

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

KIP-98 - Exactly Once Delivery and Transactional Messaging introduced EOS to kafka through transactions and idempotent producers. Idempotent producers offer guarantees like deduplication and some exactly once semantics on a single partition in a single producer session. They require only a producer ID which is assigned when the producer first starts up. Transactions offer guarantees across topic partition and producer sessions and require both a producer ID given by the system and a transactional id provided by the user.

Since transaction IDs are provided by the user and expected to persist across client restarts, we also expect the number of transactional IDs to be proportional to the number of clients. Producer IDs, when not associated to a transaction, will be created every time the producer restarts. (If they are associated to a transactional ID, the producer ID will also be reused.)

We currently store mappings from transaction ID to producer ID (along with other transactional state), and from producer ID to producer state. Both of these maps are configured to retain their values unless there are no updates to the entry for `transactional.id.expiration.ms` . Note that both maps use this configuration for expiration. The producer ID state is stored in each partition, so we could expect to see the (number of clients * the number of partitions * the number of producer restarts) number of producer IDs for idempotent producers. This is potentially a much larger number compared to the storage transactional IDs that is roughly proportional to number of clients.

With KIP-679: Producer will enable the strongest delivery guarantee by default idempotent producers became the default in Kafka. This means that unless otherwise specified, all new producers will be given producer IDs.

Some (inefficient) applications may now create many non-transactional idempotent producers. Each of these producers will be assigned a producer ID and these IDs are stored in the aforementioned producer ID to state map in ProducerStateManager. Since there may be a new influx of producer ID usage that will surpass transactional ID usage, it may be useful to have a separate timeout for producer IDs to avoid excess memory usage in this map.

Public Interfaces

We propose adding the following configuration

Name	Description	Default	Valid Values	Priority	Update Mode
`producer.id.expiration.ms`	The time in ms that a topic partition leader will wait before expiring producer ids. Producer IDs will not expire while a transaction associated to them is still ongoing. Note that producer ids may expire sooner if the last write from the producer id is deleted due to the topic's retention settings. Setting this value the same or higher than `delivery.timeout.ms` can help prevent expiration during retries and protect against message duplication, but the default should be reasonable for most use cases.	86400000 (1 day)	[1, ...]	low	per-broker

The following configuration will also be *modified*:

Name	OLD Description	New Description
`transactional.id.expiration.ms`	The time in ms that the transaction coordinator will wait without receiving any transaction status updates for the current transaction before expiring its transactional id. *This setting also influences producer id expiration - producer ids are expired once this time has elapsed after the last write with the given producer id. Note that producer ids may expire sooner if the last write from the producer id is deleted due to the topic's retention settings.*	The time in ms that the transaction coordinator will wait without receiving any transaction status updates for the current transaction before expiring its transactional id. Transactional IDs will not expire while a the transaction is still ongoing.

Proposed Changes

Replace `transactional.id.expiration.ms` with `producer.id.expiration.ms` in ProducerStateManager when checking whether producers ids should be expired. Also update the documentation for `transactional.id.expiration.ms`

Keep in mind that the expiration also checks if a transaction is in progress for a given producer ID. Thus, this change mainly impacts idempotent producers. A transactional producer won't have its ID expire during a transaction. When the transaction completes, the producer ID may be removed from the ProducerStateManager map, but will be put in the map again on the start of a new transaction.

Currently, when a idempotent producer's ID expires, it silently loses its idempotency guarantees.

NOTE: Setting `producer.id.expiration.ms` to be lower than the delivery timeout will increase the likelihood of duplicates. Any expiration of producer IDs means that the producer can no longer take advantage of idempotency guarantees.

This configuration will be dynamic. It can be updated per broker, or for the entire cluster.

Compatibility, Deprecation, and Migration Plan

New brokers will start using the new default of 1 day to expire producer IDs. In typical usage, this should not cause any noticeable difference to users since out of sequence records should be very uncommon outside of the retry window. The value can also be configured to match `transactional.id.expiration.ms` if it is necessary to keep the old behavior.

Rejected Alternatives

Have a default value of 7 days

This would be a safe option and match a configuration that did not customize `transactional.id.expiration.ms`. In the case of a custom configuration for transactional expiration, it would more easily reveal that the new configuration existed. However, since this is still not totally apparent, it is better to default to not changing current behavior.

Have a default value of -1 to use the transaction.id.expiration.ms configured value

Although this would be the most seamless transition for compatibility, it doesn't allow users to see benefits from the new configuration. The default of 1 day should not cause issues with typical clients, so it seems fair to set the default lower.

Make configuration static

Originally this configuration was created to be static. However, in the case of a misbehaving client, it may be necessary to take action to change the value. When the broker is overloading, rolling the cluster may be tricky. By having a dynamic config, we can reduce impact.