KIP-526: Reduce Producer Metadata Lookups for Large Number of Topics

Status

Current state: Under Discussion

Discussion thread: here

JIRA: KAFKA-8904

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

Motivation

When a Kafka producer sends a record to a topic, it must have metadata about the topic in order to determine the partition to which the record will be delivered. Therefore, an unknown topic's metadata must be fetched whenever it is encountered. In the producer's current incarnation, no prior knowledge of the working set of topics is provided by the client so all topic metadata is requested on-demand.

For the majority of use-cases, where a fixed and/or manageable number of topics are processed, fetching topic metadata is a cost that's incurred upon startup but subsequently mitigated by maintaining a metadata cache. However, in the case where a large or variable number of topics are produced to, clients may encounter degraded performance that severely limits processing, or in extreme degenerate cases, behavior which impedes progress altogether.

There are a couple factors in the producer that hinder client processing when working with a large number of topics:

1. The number of metadata RPCs generated.
2. The size of the metadata RPCs.
3. Metadata ages out of the cache even though it's expected to be used in the near future.

For (1), an RPC is generated every time an uncached topic's metadata must be fetched. During periods when a large number of uncached topics are processed (e.g. producer startup), a large number of RPCs may be sent out to the broker(s) in a short period of time. Generally, if there's ***n*** unknown topics, then ***O(n)*** metadata RPCs will be sent regardless to their proximity in time.

For (2), requests for metadata will also ask to refresh metadata about all known topics. As the number of topics becomes large, this will inflate the response size to be quite large and require non-trivial processing. This further exacerbates (1) in that every subsequent metadata request will result in an increasing amount of data transmitted back to the client for every RPC.

For (3), implementing (2) will reduce the cost for these occurrences. However, the duration after which metadata is evicted from the producer's metadata cache is currently a fixed value, and therefore cannot be modified the client, even if the value is too short of a duration. Implementing a way to control the eviction period should remove the need for the metadata RPC in these cases.

In concert, these factors amplify the negative effects of each other, and improvements should be made in order to alleviate topic scalability issues.

Proposed Changes

The proposal is to resolve (2) and (3), which should reduce the cost of (1) considerably.

The producer has two values of interest: an eviction threshold for topic metadata, which is used to remove an unused topic from the working set at a future time (currently hard-coded to 5 minutes), and a metadata refresh threshold, which is used to periodically refresh topic metadata (defined by `metadata.max.age.ms`). While seemingly similar, these two thresholds fundamentally differ: you could imagine a short eviction threshold in cases where records may be produced to a topic and then subsequently forgotten, or a long eviction where topics are intermittently produced to over the lifetime of the producer.

Therefore, the producer should add configuration flag '`metadata.max.idle.ms`' (default: 5 minutes) to control topic eviction.

Changes will be made to permit a subset of topics to refresh their metadata. When determining which topics' metadata to refresh, the following criteria will be used:

- If a new (uncached) topic is encountered, only fetch metadata for that particular topic. This is new.
- If a topic was notified of a metadata change (e.g. NOT_LEADER_FOR_PARTITION encountered), then update all topics in the working set.
    - The rationale is that, when such changes are encountered, it's highly probable that other topics' metadata will also need to be refreshed. This is unchanged from how the logic works today.
- If a topic's metadata that hasn't been refreshed at least '`metadata.max.age.ms`' ago, then update all topics in the working set.

    - The rationale is that, when encountered, other topics will also be nearing their metadata max age. This is unchanged from how the logic works today.

Therefore, during conditions like producer startup, only new topics' metadata will be fetched, as opposed to all topics in the working set. While it doesn't reduce the number of generated RPCs, it dramatically reduces the response payload in the worst-case, and reduces overall processing by both server and client.

Note in the event of request failures (timeouts), there is no plan to change the current behavior, which is to wait '`retry.backoff.ms`' before retrying.

Public Interfaces

Adds producer configuration flag **`metadata.max.idle.ms` `(default: 5 minutes)`** to control topic eviction duration.

```
/** <code>metadata.max.idle.ms</code> */
public static final String METADATA_MAX_IDLE_CONFIG = "metadata.max.idle.ms";
private static final String METADATA_MAX_IDLE_DOC =
        "Controls how long the producer will cache metadata for a topic that's idle. If the elapsed " +
        "time since a topic was last produced to exceeds the metadata idle duration, then the topic's " +
        "metadata is removed from the cache and the next access to it will force a metadata fetch request.";

...
    .define(METADATA_MAX_IDLE_CONFIG,
            Type.LONG,
            5 * 60 * 1000,
            atLeast(5000),
            Importance.LOW,
            METADATA_MAX_IDLE_DOC)
```

Compatibility, Deprecation, and Migration Plan

Impact on client will be strictly internal performance improvements; no public APIs, protocols, or other external factors are being changed.

Rejected Alternatives

- Allow the producer to specify interested topics. In doing so, many uncached topics could be fetched in a single request before records were produced for them, e.g. at startup. This would greatly alleviate the problem, however requires the clients to (1) implement a new producer call, and (2) know the working set of topics a priori. It'd obviate the need for fetching metadata asynchronously, which wouldn't resolve the throughput "bubble" that individual producer threads encounter when waiting for new topic metadata to be fetched.