KIP-952: Regenerate segment-aligned producer snapshots when upgrading to a Kafka version supporting Tiered Storage

Authors: Mital Awachat Christo Lolov

Status

Current state: *"Under Discussion"*

Discussion thread: here

JIRA: KAFKA-15195

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

Motivation

Our motivation is to provide an upgrade path for customers with Kafka versions < 2.8.0 wanting to enable Tiered Storage in 3.6.

KIP-405: Kafka Tiered Storage provides infinite storage to Kafka. To achieve this Kafka uploads segments, indices, leader epochs and producer snapshots to remote storage. Producer snapshots were aligned to segments as part of addressing

Unable to render Jira issues macro, execution error.

in Apache Kafka 2.8.0. **As mentioned in KIP-405: Kafka Tiered Storage#Upgrade a customer wishing to upgrade from a Kafka version < 2.8.0 to 3.6 and turn Tiered Storage on will have to wait for retention to clean up segments without an associated producer snapshot.**

However, in our experience, customers of Kafka expect to be able to immediately enable tiering on a topic once their cluster upgrade is complete. Once they do this, however, they start seeing NPEs and no data is uploaded to Tiered Storage (https://github.com/apache/kafka/blob/9e50f7cdd37f923cfef4711cf11c1c5271a0a6c7/storage/api/src/main/java/org/apache/kafka/server/log/remote/storage/LogSegmentData.java#L61).

To achieve this, we propose changing Kafka to retroactively create producer snapshot files on upload whenever a segment is due to be archived and lacks one.

Public Interfaces

Briefly list any new interfaces that will be introduced as part of this proposal or any existing interfaces that will be removed or changed. The purpose of this section is to concisely call out the public contract that will come along with this feature.

No public interfaces will be changed as part of this KIP.

Proposed Changes

Describe the new thing you want to do in appropriate detail. This may be fairly extensive and have large subsections of its own. Or it may be a few sentences. Use judgement based on the scope of the change.

Background of ProducerIds

The ProducerStateManager is a logical component of Kafka which keeps a map from **producer identifiers** to the **last record written by that producer**. T here is a separate ProducerStateManager for each partition. Every time a segment is rolled this map is written to a producer snapshot file (since 2.8.0). The state is also written to a snapshot file on shutdown. These snapshot files are used to recover the broker faster on startup. Much like the index and timeindex files the state stored in the producer snapshots can be recreated from the segments. Unlike the index and timeindex files, however, they require the **whole log to be replayed** from the last available snapshot or if none are available from the beginning of the partition.

If a producer identifier is not present in the ProducerStateManager when it tries to write records it will receive an UNKNOWN_PRODUCER_ID error. The producer will register a new identifier with the broker and from then on it will write records as normal. Idempotency is ensured for a producer which keeps the same producer identifier, however it is not ensured if the producer changes their identifier.

Proposed solution

There is a configuration called `producer.id.expiration.ms` whose default value is 1 day (86400000 ms). In theory, if we have an infinite log and we have lost all producer snapshots we should start snapshot recovery from segments no older than 1 day. All producer identifiers from earlier segments would have expired and a producer trying to write records with such an identifier would be asked to request a new one.

As such, the ideal solution when trying to upload a segment to Tiered Storage and not encountering a snapshot would be recreate it by replaying segments no older than the value of `producer.id.expiration.ms.`

Compatibility, Deprecation, and Migration Plan

- *What impact (if any) will there be on existing users?* N/A
- *If we are changing behavior how will we phase out the older behavior?* N/A
- *If we need special migration tools, describe them here.* N/A
- *When will we remove the existing behavior?* N/A

Test Plan

Describe in few sentences how the KIP will be tested. We are mostly interested in system tests (since unit-tests are specific to implementation details). How will we know that the implementation works as expected? How will we know nothing broke?

Unit and integration tests.

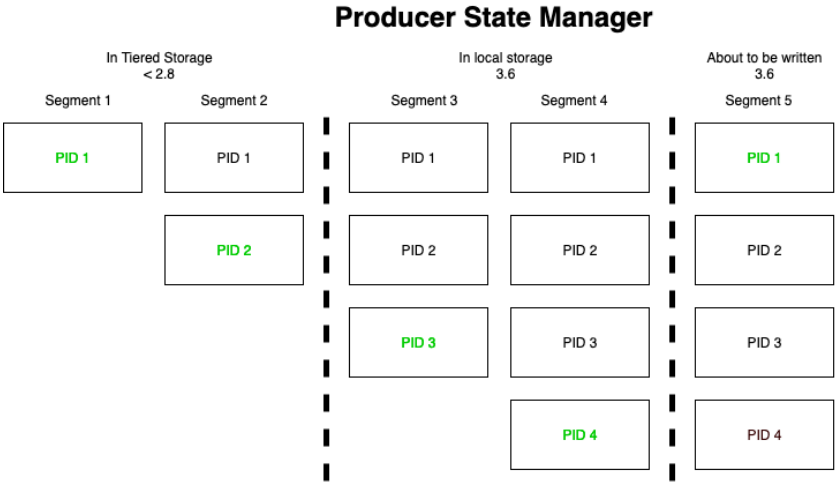
Rejected Alternatives

If there are alternative ways of accomplishing the same thing, what were they? The purpose of this section is to motivate why the design is the way it is and not some other way.

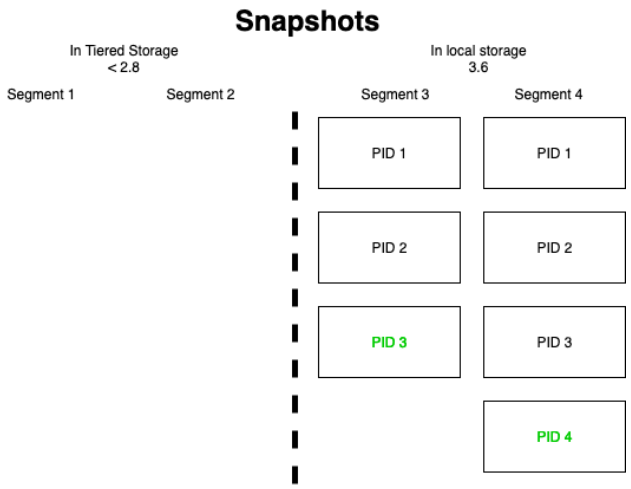
1. Create an empty snapshot file on write if a snapshot file is not found
   Create a segment-aligned empty producer snapshot whenever we detect that the value here is null. The rest of Tiered Storage's logic should then pick the file up for upload with no changes. No changes will be required on the read path.

   - Is adding an empty snapshot file safe? If not, what do we risk?

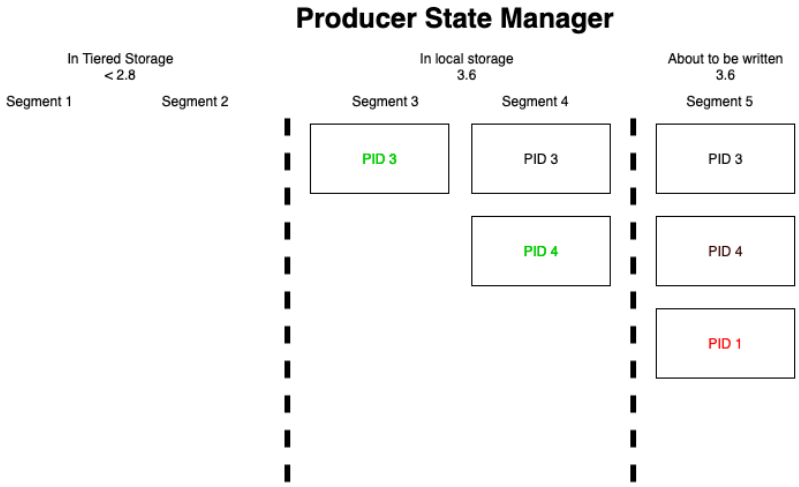
   Encountering an empty snapshot file means that that particular Kafka broker will treat all idempotent producers who produced prior to the offset of the snapshot file as expired. In the situation where a follower broker has rebuilt the auxiliary state from an empty snapshot, the leader broker has just gone down, some producer identifiers have not produced since the offset of the empty snapshot, and the follower broker will become the leader, then those producer identifiers will be expired potentially resulting in duplicate records. Given that after the upgrade from < 2.8 to 3.6 segments will start having an associated snapshot the above situation also needs to happen for segments on the boundary between the two versions. Here is the same example illustrated:



This figure shows the true state in the Producer State Manager on the leader broker which has never crashed. In each segment a new producer has produced (green) up until the about to be written one where we know that producer 1 would write again the same record it tried writing in segment 1.



This figure shows what snapshots the leader broker would have uploaded to Tiered Storage. Since segments 1 and 2 have been created prior to Kafka 2.8 they don't have "true" associated snapshot files. As such, when they are tiered empty segment-aligned producer snapshots containing no producer identifiers are written. On the leader broker, there would be snapshots written for segments 3 and 4 containing the correct state from the Producer State Manager and featuring all producer identifiers.



This figure shows the Producer State Manager of a follower broker which has come online after Kafka was upgraded to 3.6 and it contained no local segments. It obtained the empty snapshot file from segment 2 and after replicating from the leader it learned about producer identifiers 3 and 4. Right before segment 5 was written to by producer 1 the leader broker dies, transfers leadership to the follower broker and the follower broker receives a record from producer 1. This broker is unaware of this producer identifier, considers it expired and asks the producer to create a new identifier (red). Once the producer has done this it is allowed to write the same record it wrote in segment 1 thus resulting in a duplicate.

Let us assume a per segment write throughput of 5 MB/s, a default `log.segment.bytes` of 1 GB and a default `producer.id.expiration.ms` of 1 day. A new segment is rolled every 1 * 1024 / 5 ~ 204 seconds ~ 3.5 minutes. For the above situation to occur a producer needs to not have written for ~15 minutes at the boundary between < 2.8 and 3.6 to result in duplicates.

2. Create empty snapshot files on read if a snapshot file is not found - it would not be immediately obvious whether a snapshot is not presented because of an upgrade of Kafka version, because there is a bug in the plugin implementation or because the remote storage itself has been tampered with - if we explicitly write an empty file we reduce some of this ambiguity. This would also suffer from the same problem as rejected alternative number 1.
3. Make Kafka not upload segments to remote storage until it has expired all segments lacking a producer snapshot - we think this is just as unexpected (especially if customers aren't well-aware of what producer snapshots are used for) as what the current approach is.
4. Do not allow a topic to have its tiering configuration set to true until all of its segments have an associated producer snapshot - the configuration will be forwarded and accepted by the controller - this means that the controller needs to know whether the current partition leader has expired all segments from Kafka versions < 2.8 which while theoretically possible will not be straightforward to implement.
5. Do not archive snapshot files - like indices, snapshot files can be recreated by rereading the log. However, in the case of Tiered Storage we make the assumption that replaying the whole log will be quite costly.