KIP-424: Allow suppression of intermediate events based on wall clock time

Status

Current state: Under Discussion

Discussion thread: KIP-424: Allow suppression of intermediate events based on wall clock time

JIRA: KAFKA-7748

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

Motivation

Currently, Kafka Streams offers the ability to suppress updates for KTables using either RecordTime or WindowEndTime, which are in turn defined by stream time. This capability is discussed in detail in KIP-328: Ability to suppress updates for KTables, specifically section 2a: **In termediate Events How long to wait for more updates before emitting.** It would be helpful to have another option that would allow suppression of intermediate events based on wall clock time. This would allow a user to cap the rate that aggregates are produced independent of their stream time.

Public Interfaces

This proposal would add a new method, `untilWallClockTimeLimit`, to the `Suppressed` interface:

```
/**
 * Configure the suppression to wait {@code timeToWaitForMoreEvents} amount of wall clock time after receiving a
record
 * before emitting it further downstream. If another record for the same key arrives in the mean time, it replaces
 * the first record in the buffer but does <em>not</em> re-start the timer.
 *
 * @param timeToWaitForMoreEvents The amount of wall clock time to wait, per record, for new events.
 * @param bufferConfig A configuration specifying how much space to use for buffering intermediate results.
 * @param <K> The key type for the KTable to apply this suppression to.
 * @return a suppression configuration
 */
static <K> Suppressed<K> untilWallClockTimeLimit(final Duration timeToWaitForMoreEvents, final BufferConfig
bufferConfig);
```

Proposed Changes

The intention is to configure an upper bound for how frequently streams will send updates for a particular windowed key downstream. It will function as a rate limiter for cases where the cache fills up, or there is no cache, to ensure streams will not flood downstream. It does require data to flow through at at least the rate specified in order to maintain that flow. For example, configuring a `timeToWaitForMoreEvents` for a duration of 5 seconds would mean that you never receive more than one update downstream for a key within any 5 second period. However, checks against the wall clock are event driven: if no events are received in over 5 seconds, no events will be sent downstream.

The main code change here is in `KTableSuppressProcessor.enforceConstraints()`, we will need a new predicate to determine when to emit. The `InMemoryTimeOrderedKeyValueBuffer` may need to keep track of when we last evicted.

Compatibility, Deprecation, and Migration Plan

- This shouldn't pose any issues for existing users as it's a feature enhancement.

Rejected Alternatives

The `commit.interval.ms` configuration variable provides a helpful upper bound in how often updates are emitted downstream. However, this requires that `cache.max.bytes.buffering` is never full, which is difficult to manage for certain loads and defeats the purpose of having RocksDB as a state store.