KIP-539: Implement mechanism to flush out records in low traffic suppression buffers

Status

Current state: *Under Discussion*

Discussion thread: *here*

JIRA: *here*

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

Motivation

Currently, in Kafka Streams, if a suppression buffer has low traffic for the number of records received (i.e. the buffer could not advance the stream time which would cause records to get stuck in the buffers). Consequently, records would remain in these buffers unless further records were received and the stream time be advanced. Therefore, we need a way to flush out these records such the user can access them despite the low traffic constraint. This is also a future step towards tracking stream time independently on a per key basis rather than per partition.

This situation is in fact worse for certain cases where sensors would wait for 24 hours before dumping data into a stream. So realistically, we could end up in a situation where suppress processors does not receive records for long periods of time (on the scale of hours) and that means that stream time cannot be advanced, thus no records are ever emitted during this time of inactivity. For most users, they wish to avoid this situation, so we should work to resolve this by flushing out records every so often to prevent them from getting "stuck" in the buffer.

By extension, we could also implement a feature that has been requested in the past: allow suppressed operators to suppress records based on wall-clock time.

Public Interfaces

This KIP would add the following method 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

All things considered, we should automate the flushing of records that have remained in the buffers for too long. For many Kafka users, they prefer not to have to manually call a method which would trigger such an operation. In light of such an opinion, the best approach would be to add a config `suppressed.record.retention.ms` which would be defined by the user to indicate how long they wish a record to remain in the suppression buffer.

However, most if not all processors in Kafka as of the moment does not have a `Time` instance as a field in its implementations (or an internal clock). Thus, they cannot track time (not stream time but how much time has passed IRL). There is a way around this though. Periodically, what we could let the `StreamThread` do is insert a record (similar to a tombstone, kind of like a null record) which would be given to the processor to process. We don't have to check if a processor is explicitly a suppress processor (which means checking if types match). Instead, what we could do is add a method to `InternalProcessorContext` called `insertNullRecord` which by default does nothing (hence it is empty).

Specifically, for suppress processors, we could override this method and have `insertNullRecord` call the actual `process()` method with a null key and value (with `time.milliseconds()` being the record time). When the suppress processor receives this record, it will retain a timestamp field which stores when the last time a null record was inserted. In this manner, we are able to track roughly how much time has passed within a suppress processor without having to insert an actual `Time` instance as field.

In this manner, we could then expire records which have exceeded `suppressed.record.retention.ms` and evict the records as we see fit.

Please note that using this approach, the amount of time that has passed in a suppress processor is approximated (reasonably well). If we really need to, we might need to add another config to determine how often we will send these tombstone records (please note that they are never exposed to the user, as it is completely hidden within one processor).

Rejected Alternatives

There has been some thoughts of adding a `Time` instance to suppress processors, but that is against convention since processors should not keep track of time (since it is also bloody expensive). Instead, that should be left to stream threads and other processes.