KIP-274: Kafka Streams Skipped Records Metrics

Status

Current state: *Adopted*

Discussion thread: http://mail-archives.apache.org/mod_mbox/kafka-dev/201803.mbox/browser

JIRA: **KAFKA-6376** - Getting issue details... STATUS

PR: https://github.com/apache/kafka/pull/4812

Released: 2.0

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

Motivation

Operators of Kafka Streams applications need awareness of records being skipped.

We currently present these metrics:

- From the StreamThread:
    - `MetricName[name=skipped-records-total, group=stream-metrics, description=The total number of skipped records, tags={client-id=...(per-thread client-id)}] (INFO level)`
- From the ProcessorNode:
    - `MetricName[name=skippedDueToDeserializationError-rate, group=stream-processor-node-metrics, description=The average number of occurrence of skippedDueToDeserializationError operation per second., tags={task-id=..., processor-node-id=...}] (DEBUG level)`
    - `MetricName[name=skippedDueToDeserializationError-total, group=stream-processor-node-metrics, description=The total number of occurrence of skippedDueToDeserializationError operations., tags={task-id=..., processor-node-id=...}] (DEBUG level)`

This arrangement is unsatisfying for two reasons:

1. The thread-level metric actually does not capture "the total number of skipped records", but only a subset of them. Also, 'skippedDueToDeserializationError' will be incremented both for deserialization errors and for negative timestamps. Technically speaking, these are bugs and not KIP-worthy, but I'm going to propose restructuring the metrics to make such bugs less likely in the future.
2. There are more sources of skipped records than deserialization errors. This means we need new metrics, which **is** KIP-worthy.

A third motivation is this:

We provide the TopologyTestDriver for testing streams topologies. An application author may wish to inspect the metrics after a test, but since the driver does not construct a StreamThread, the aggregated metric is unavailable. The two ProcessorNode metrics are available. However, it's currently not possible to access the metrics from TopologyTestDriver at all.

Proposed Public Interface Change

Restructuring metrics around skipped records

I propose to remove the other skipped-record metrics and only the following metrics at the StreamThread level:

```
MetricName[name=skipped-records-rate, group=stream-metrics, description=The total number of skipped records, tags={client-id=...(per-thread client-id)}]
```

```
MetricName[name=skipped-records-total, group=stream-metrics, description=The average number of skipped records per second, tags={client-id=...(per-thread client-id)}]
```

All of these metrics would be INFO level.

Instead of also maintaining more DEBUG-level granular skipped metrics (such as "skippedDuetoDeserializationError"), we will capture useful details about the record that got skipped (topic, partition, offset) as well as the reason for the skip ("deserialization error", "negative timestamp", etc.) with a WARN level log.

This provides a slim an clear interface (only one metric to monitor), while still exposing the granular information for those who need to track down the reason for the skips. In fact, having log messages for debugging is better for this case, since we can include contextual information about the record that got skipped. And this is also a common division of responsibility between the two information sources: metrics are for monitoring, logs are for debugging.

TopologyTestDriver#metrics()

I propose to add the following method to the TopologyTestDriver:

```
public Map<MetricName, ? extends Metric> metrics()
```

This is the same interface presented by KafkaStreams#metrics(). As TopologyTestDriver is otherwise analogous to KafkaStreams, this addition to the interface makes sense, and it would allow access to the metrics from test code.

Compatibility, Deprecation, and Migration Plan

We have a choice whether to remove or deprecate the existing metrics. I have no strong opinion, although I lean more heavily toward immediate removal.

Since there are no compiler warnings, I think it's likely that "deprecating" the existing metrics will go unnoticed, and the metrics disappearing would be just as much a surprise at the end of the deprecation period as it would be immediately.

The deprecation period would actually just create a period of ambiguity in which both old and new metrics are available, making things more confusing for operators and also increasing the changes that people would couple to the wrong (deprecated) metrics.

Test Plan

We would create a new test probing the various reasons for skipping records and verifying the metrics reflect them.

Rejected Alternatives

We could instead just make the minimal change of correcting the acknowledged bugs and adding the missing measurement points. But I think that we have an opportunity to improve the ergonomics of the metrics for operators while also addressing the current issues.

We also discussed keeping one metric per skip reason, but that felt too granular. Plus, having the metrics set to DEBUG creates a visibility problem: A) People can't discover the metric just by looking through the available metrics at run time, B) Assuming there's an unexpected skip reported, operators may turn on DEBUG, but can't be guaranteed to see another skip. Logging the granular information is the alternative we settled on.

We discussed including some built-in rollup mechanism to produce aggregated metrics at the top level. This is difficult for us to do well from within the Streams library, since it would involve each instance observing the metrics of all the other instances. We could create some kind of tool to do metric aggregation, but it just seems out of scope for this KIP. If metric aggregation is a general problem the community surfaces, we'll tackle it with a separate KIP. This decision allows us to focus in the library about surfacing the right information at the right level of granularity.

We discussed moving the skipped-records metric to the StreamTask. This has the advantage that they will automatically be included in the TopologyTestDriver, but the disadvantage that they increase the total number of metrics to pay attention to. The StreamThread is the highest level of summary that we can provide without introducing lock contention while updating the metrics.