KIP-105: Addition of Recording Level for Sensors

- Status
- Motivation
- Public Interfaces
- Proposed Changes
- Rejected Alternatives
- Future work

.

Status

Current state: Accepted

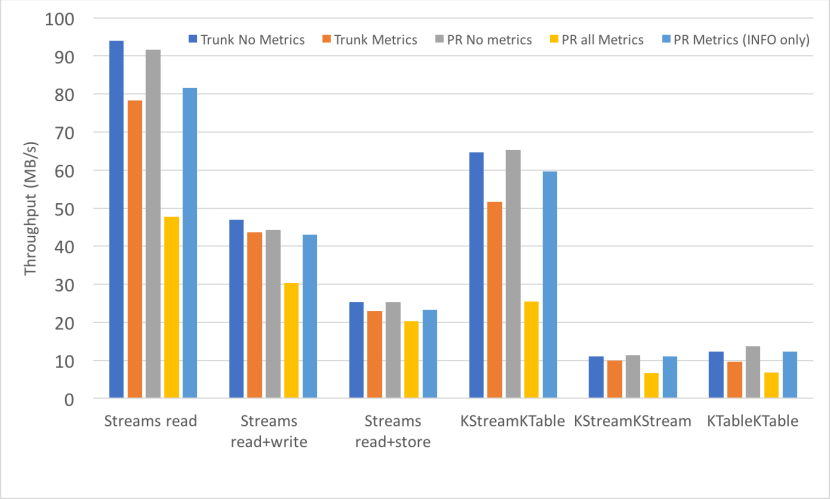
Discussion thread: *here*

JIRA: *KAFKA-3715 Higher granularity streams metrics*

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

Motivation

- Recording metrics is expensive. There are scenarios where, for debug purposes, tens or hundreds of metrics need to be recorded. The performance impact of always recording can be as high as 50% at times (see graph below showing some Kafka Streams microbenchmarks. This graph is also part of PR for the JIRA for this KIP. Compare the yellow bar which is with all metrics enabled, with any other bar). We would like a way to control which metrics are recorded to address this performance problem.



- This KIP proposes the addition of a RecordingLevel in Sensor.java. This associates each sensor with a recording level and the metrics and computations associated with the sensor are recorded or calculated only if the metric config of the client requires these metrics to be recorded. For example, a sensor associated with a RecordingLevel.DEBUG recording level will only record metrics, (and correspondingly compute the time. milliseconds or nanoseconds if required by its metrics), if the client config for metric recording levels is set to DEBUG. Note that the sensor will still be registered and polled by any reporters, however the expensive part of updating its metrics will not be done. This provides a mechanism for clients to toggle metric granularity without a change to individual sensors, and we do not perform unnecessary intensive computations associated on metrics which must not be recorded.
- An objective of this KIP is to leave open the possibility of changing the recording level at runtime, although it is not in the scope of this KIP to implement that right now.

Public Interfaces

- Changes to public class Sensor.java to include Sensor.RecordingLevel
- Addition of static configuration string in CommonClientConfigs.java
- Exposing that configuration string in StreamsConfig.java and KafkaConfig.scala since both use CommonClientConfigs.

Proposed Changes

- The constructor of Sensor.java now takes a Sensor.RecordingLevel as a parameter. This is an enum which has DEBUG and INFO recording levels, and can be extended in future to include other levels.

  Sensor(Metrics registry, String name, Sensor[] parents, MetricConfig config, Time time,
  long inactiveSensorExpirationTimeSeconds, RecordingLevel recordingLevel)

- The ClientConfig will have a new option called 'metrics.recording.level' which is set to match the recording levels of INFO or DEBUG while defining the consumer config.
- In Sensor.java the shouldRecord() method is used to compare the value of metric recording level in the consumer config and the RecordingLevel associated with the sensor, to determine if metrics should recorded.

  From Sensor.java

```
 /**
 * @return true if the sensor's recording level indicates that the metric will be recorded, false otherwise
 */
public boolean shouldRecord() {
    return this.recordingLevel.shouldRecord(config.recordingLevel());
}
```

  From nested enum, Sensor.RecordingLevel

```
public boolean shouldRecord(final RecordingLevel configLevel) {
    if (configLevel.equals(DEBUG)) {
        return true;
 } else {
        return configLevel.equals(this);
 }
}
```

- Further, since both StreamsConfig.java and KafkaConfig.scala use CommonClientConfigs, we also expose that field in both classes respectively.
- Note that any MetricReporter implementations have access to the recording level via KafkaMetric.config().recordingLevel(). Furthermore, the MetricReporters can retrieve the active config for the recording level via the configure method.

Compatibility, Deprecation, and Migration Plan

- no migration plan needed, backward compatible constructor for Sensor.java which adds RecordingLevel.INFO where a recording level is not specified.

Rejected Alternatives

- We considered whether we want to register a DEBUG sensor at all if the config is INFO. Currently we do register it, but we do not record any values. This was rejected because it would add complexity to Sensor.java. A client could attempt to register a metric and report a metric against a sensor which is not internally registered.
- Similarly, we could register a DEBUG sensor if the config is INFO, but have the reporter not poll it. Again this would add some complexity that might not be warranted since the expensive part this KIP tries to address is the management of metrics associated with a sensor.

Future work

- The KIP focuses on one problem only, which is the overhead of computing metrics. It allows for a subsequent JIRA to have a way to change the levels at runtime via JMX. It also allows for a subsequent JIRA to provide more tags to the metrics reporter if desired (e.g., "debug", "info") so that the user has a debug or info dashboard. In other words, the built in JMX reporter will not make use of the recording level info in this KIP.
- It may be useful to allow specifying different recording levels for different hierarchies of sensors (similar to logging levels for different loggers).