

# KIP-471: Expose RocksDB Metrics in Kafka Streams


- [Status](#)
- [Motivation](#)
- [Public Interfaces](#)
- [Proposed Changes](#)
  - [Metrics to Add](#)
    - [bytes-written-\(rate|total\)](#)
    - [bytes-read-\(rate|total\)](#)
    - [memtable-bytes-flushed-\(rate|total\)](#) and [memtable-flush-time-\(avg|min|max\)](#)
    - [memtable-hit-ratio](#)
    - [block-cache-data-hit-ratio](#), [block-cache-index-hit-ratio](#), and [block-cache-filter-hit-ratio](#)
    - [bytes-read-compaction-rate](#), [bytes-written-compaction-rate](#), and [compaction-time-\(avg|min|max\)](#)
    - [write-stall-duration\(avg|total\)](#)
    - [num-open-files](#) and [num-file-errors-total](#)
  - [Metrics for States Consisting of Multiple RocksDB Instances](#)
    - [Rates](#)
    - [Hit Ratios](#)
    - [Totals](#)
    - [Averages](#)
    - [Minima](#)
    - [Maxima](#)
    - [num-open-files](#)
- [Compatibility, Deprecation, and Migration Plan](#)
- [Rejected Alternatives](#)

## Status

**Current state:** Accepted

**Discussion thread:** [here](#)

JIRA:

 Unable to render Jira issues macro, execution error.

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

## Motivation

RocksDB has functionality to collect statistics about its operations to monitor running RocksDB's instances. These statistics enable users to find bottlenecks and to accordingly tune RocksDB. RocksDB's statistics can be accessed programmatically via JNI or RocksDB can be configured to periodically dump them to disk. Although RocksDB provides this functionality, Kafka Streams does currently not expose RocksDB's statistics in its metrics. Hence users need to implement Streams' RocksDBConfigSetter to fetch the statistics. This KIP proposes to expose a subset of RocksDB's statistics in the metrics of Kafka Streams.

## Public Interfaces

Each exposed metric will have the following tags:

- type = stream-state-metrics,
- thread-id = [thread ID],
- task-id = [task ID]
- rocksdb-state-id = [store ID] for key-value stores
- rocksdb-session-state-id = [store ID] for session stores
- rocksdb-window-state-id = [store ID] for window stores

The following metrics will be exposed in the Kafka Streams' metrics

- bytes-written-rate [bytes/s]
- bytes-written-total [bytes]
- bytes-read-rate [bytes/s]
- bytes-read-total [bytes]

- memtable-bytes-flushed-rate [bytes/s]
- memtable-bytes-flushed-total [bytes]
- memtable-flush-time-(avg|min|max) [ms]
- memtable-hit-ratio
- block-cache-data-hit-ratio
- block-cache-index-hit-ratio
- block-cache-filter-hit-ratio
- bytes-read-compaction-rate [bytes/s]
- bytes-written-compaction-rate [bytes/s]
- compaction-time-(avg|min|max) [ms]
- write-stall-duration-(avg|total) [ms]
- number-open-files
- number-file-errors-total

The recording level for all metrics will be DEBUG.

## Proposed Changes

In this section, I will explain the meaning of the metrics listed in the previous section and why I chose them. Generally, I tried to choose the metrics that are useful independently of any specific configuration of the RocksDB instances. Furthermore, I tried to keep the number of metrics at a minimum, because adding metrics in future is easier than deleting them from a backward-compatibility point of view. Finally, I will explain how to compute metrics for state stores that consist of multiple RocksDB instances, e.g., window and session stores.

### Metrics to Add

#### **bytes-written-(rate|total)**

These metrics measure the bytes written to a RocksDB instance. The metrics show the write load on a RocksDB instance.

#### **bytes-read-(rate|total)**

Analogously to bytes-written-(rate|total), these metrics measure the bytes read from a RocksDB instance. The metrics show the read load on a RocksDB instance.

#### **memtable-bytes-flushed-(rate|total) and memtable-flush-time-(avg|min|max)**

When data is put into RocksDB, the data is written into a in-memory tree data structure called memtable. When the memtable is almost full, data in the memtable is flushed to disk by a background process. Metrics bytes-flushed-(rate|total) measure the average throughput of flushes and the total amount of bytes written to disk. Metrics flush-time-(avg|min|max) measure the processing time of flushes.

The metrics should help to identify flushes as bottlenecks.

#### **memtable-hit-ratio**

When data is read from RocksDB, the memtable is consulted firstly to find the data. This metric measures the number of hits with respect to the number of all lookups into the memtable. Hence, the formula for this metric is  $\text{hits}/(\text{hits} + \text{misses})$ .

A low memtable-hit-ratio might indicate a too small memtable.

#### **block-cache-data-hit-ratio, block-cache-index-hit-ratio, and block-cache-filter-hit-ratio**

If data is not found in the memtable, the block cache is consulted. Metric block-cache-data-hit-ratio measures the number of hits for data blocks with respect to the number of all lookups for data blocks into the block cache. The formula for this metric is the equivalent to the one for memtable-hit-ratio.

Metrics block-cache-index-hit-ratio and block-cache-filter-hit-ratio measure the hit ratio for index and filter blocks if they are cached in the block cache. By default index and filter blocks are cached outside of block cache. Users can configure RocksDB to include index and filter blocks into the block cache to better control the memory consumption of RocksDB. If users do not opt to cache index and filter blocks in the block cache, the value of these metrics should stay at zero.

A low hit-ratio might indicate a too small block cache.

#### **bytes-read-compaction-rate, bytes-written-compaction-rate, and compaction-time-(avg|min|max)**

After data is flushed to disk, the data needs to be reorganised on disk from time to time. This reorganisation is called compaction and is performed by a background process. For the reorganisation, the data needs to be moved from disk to memory and back. Metrics bytes-read-compaction-rate and bytes-written-compaction-rate measure read and write throughput of compactions on average. Metrics compaction-time-(avg|min|max) measure the processing time of compactions.

The metrics should help to identify compactions as bottlenecks.

#### **write-stall-duration(avg|total)**

As explained above, from time to time RocksDB flushes data from the memtable to disk and reorganises data on the disk with compactions. Flushes and compactions might stall writes to the database, hence the writes need to wait until these processes finish. These metrics measure the average and total duration of write stalls.

If flush and compaction happen too often and stall writes this time will increase and signal a bottleneck.

## num-open-files and num-file-errors-total

Part of the data in RocksDB is kept in files. These files need to be opened and closed. Metric num-open-files measures the number of currently open files and metric num-file-errors-total measures the number of file errors. Both metrics may help to find issues connected to OS and file systems.

## Metrics for States Consisting of Multiple RocksDB Instances

A state store shown in the topology description is a logical state store. Each logical state store might consist of one or multiple physical state stores, i.e., the actual state stores instances that hold the data of a logical state store. For example, window and session stores are implemented as segmented stores, i.e., each store consists of multiple segments. For persistent segmented stores, each segment is a distinct physical store and each physical store is a distinct RocksDB instance. While the fact that some logical state stores consist of multiple physical state stores is an implementation detail, it is still important for the sake of documentation to specify how metrics for such state stores are exposed and computed.

First of all, I propose to expose RocksDB metrics for each logical state store that contains one or multiple physical RocksDB instances. That is, there will be just one set of the above mentioned metrics for each logical state store and not one set for each physical RocksDB instance. Hence, the values of the tags rocksdb-(window|session)-state-id will only contain the common prefix of all physical RocksDB instances belonging to one logical state store. Furthermore, the metrics need to be aggregated over all physical RocksDB instances belonging to the same logical state store. How to aggregate the above metrics over multiple RocksDB instances is specified in the following ( $\mathcal{I}$  is the set of RocksDB instances per logical state store):

### Rates

For recorded metrics values in a sample,  $\text{metric-rate} = \frac{\sum_{i \in \mathcal{I}} \text{metric}_{_i}}{\text{time interval of sample}}$

Sampling functionality is provided by Kafka's `Sensors`.

Affected metrics: bytes-written-rate, bytes-read-rate, memtable-bytes-flushed-rate, bytes-read-compaction-rate, bytes-written-compaction-rate

### Hit Ratios

$\text{metric-hit-ratio} = \frac{\sum_{i \in \mathcal{I}} \text{hits}_{_i}}{\sum_{i \in \mathcal{I}} \text{hits}_{_i} + \sum_{i \in \mathcal{I}} \text{misses}_{_i}}$

Affected metrics: memtable-hit-ratio, block-cache-data-hit-ratio, block-cache-index-hit-ratio, block-cache-filter-hit-ratio

### Totals

$\text{metric-total} = \sum_{i \in \mathcal{I}} \text{metric}_{_i}$

Affected metrics: bytes-written-total, bytes-read-total, memtable-bytes-flushed-total, write-stall-duration-total, num-file-errors-total

### Averages

$\text{metric-avg} = \frac{\sum_{i \in \mathcal{I}} \text{metric-sum}_{_i}}{\sum_{i \in \mathcal{I}} \text{metric-count}_{_i}}$

For each average metric provided by RocksDB, there are also corresponding sum and count metrics.

Affected metrics: memtable-flush-time-avg, compaction-time-avg, write-stall-duration-avg

### Minima

$\text{metric-min} = \min_{i \in \mathcal{I}} (\text{metric-min}_{_i})$

Affected metrics: memtable-flush-time-min, compaction-time-min

### Maxima

$\text{metric-max} = \max_{i \in \mathcal{I}} (\text{metric-max}_{_i})$

Affected metrics: memtable-flush-time-max, compaction-time-max

## num-open-files

$\text{num-open-files} = \sum_{i \in \mathcal{I}} \text{num-open-files}_{_i}$

## Compatibility, Deprecation, and Migration Plan

Since metrics are only added and no other metrics are modified, this KIP should not

- affect backward-compatibility
- deprecate public interfaces
- need a migration plan other than adding the new metrics to its own monitoring component

## Rejected Alternatives

- Metrics bytes-read-compaction-total and bytes-written-compaction-total did not seem useful to me since they would measure bytes moved between memory and disk due to compaction. The metric bytes-flushed-total gives at least a feeling about the size of the persisted data in the RocksDB instance.
- Providing metrics for each physical state store was rejected since it would expose implementation details like segmented state stores.
- Providing metrics only for the hottest physical state store of a logical state store was rejected since it would not reflect the real load on the logical state store. For example, the load of interactive queries is hard to predict and this load could hit any of the physical stores not just the hottest.