7.2.12

Apache Kudu Background Operations

Date published: 2020-11-04
Date modified: 2021-10-25

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Contents

Maintenance manager

The maintenance manager schedules and runs background tasks. At any given point in time, the maintenance manager is prioritizing the next task based on improvements needed at that moment, such as relieving memory pressure, improving read performance, or freeing up disk space.

The number of worker threads dedicated to running background tasks can be controlled by setting --maintenance_manager_num_threads.

With Kudu 1.4, the maintenance manager features improved utilization of the configured maintenance threads. Previously, maintenance work would only be scheduled a maximum of 4 times per second, but now maintenance work will be scheduled immediately whenever any configured thread is available. Make sure that the --maintenance_manager_num_threads property is set to at most a 1:3 ratio for Maintenance Manager threads to the number of data directories (for spinning disks). This will improve the throughput of write-heavy workloads.

Flushing data to disk

Flushing data from memory to disk relieves memory pressure and can improve read performance by switching from a write-optimized, row-oriented in-memory format in the MemRowSet, to a read-optimized, column-oriented format on disk.

Background tasks that flush data include FlushMRSOp and FlushDeltaMemStoresOp. The metrics associated with these operations have the prefix flush_mrs and flush_dms, respectively.

With Kudu 1.4, the maintenance manager aggressively schedules flushes of in-memory data when memory consumption crosses 60 percent of the configured process-wide memory limit. The backpressure mechanism which begins to throttle client writes was also adjusted to not begin throttling until memory consumption reaches 80 percent of the configured limit. These two changes together result in improved write throughput, more consistent latency, and fewer timeouts due to memory exhaustion.

Compacting on-disk data

Kudu constantly performs several compaction tasks in order to maintain consistent read and write performance over time.

- A merging compaction, which combines multiple DiskRowSets together into a single DiskRowSet, is run by CompactRowSetsOp.
- Kudu also runs two types of delta store compaction operations: MinorDeltaCompactionOp and MajorDeltaCompactionOp.

  For more information on what these compaction operations do, see the Kudu Tablet design document.

The metrics associated with these tasks have the prefix compact_rs, delta_minor_compact_rs, and delta_major_compact_rs, respectively.

Write-ahead log garbage collection

Kudu maintains a write-ahead log (WAL) per tablet that is split into discrete fixed-size segments. A tablet periodically rolls the WAL to a new log segment when the active segment reaches a size threshold (configured by the --log_segment_size_mb property).

In order to save disk space and decrease startup time, a background task called LogGCOp attempts to garbage-collect (GC) old WAL segments by deleting them from disk once it is determined that they are no longer needed by the local node for durability.

The metrics associated with this background task have the prefix log_gc.

Tablet history garbage collection and the ancient history mark

Kudu uses a multiversion concurrency control (MVCC) mechanism to ensure that snapshot scans can proceed isolated from new changes to a table. Therefore, periodically, old historical data should be garbage-collected (removed) to free up disk space. While Kudu never removes rows or data that are visible in the latest version of the data, Kudu does remove records of old changes that are no longer visible.

The specific threshold in time (in the past) beyond which historical MVCC data becomes inaccessible and is free to be deleted is called the ancient history mark (AHM). The AHM can be configured by setting the --tablet_history_max_age_sec property.

There are two background tasks that remove historical MVCC data older than the AHM:

- The one that runs the merging compaction, called CompactRowSetsOp (see above).
- A separate background task deletes old undo delta blocks, called UndoDeltaBlockGCOp. Running UndoDeltaBlockGCOp reduces disk space usage in all workloads, but particularly in those with a higher volume of updates or upserts. The metrics associated with this background task have the prefix undo_delta_block.