Item 57
git_comments:

1. for testing
2. this is good, do nothing
3. this is good

git_commits:

1. **summary:** KAFKA-7023: Add unit test (#5197)
   message: KAFKA-7023: Add unit test (#5197) Add a unit test that validates after restoreStart, the options are set with bulk loading configs; and after restoreEnd, it resumes to the customized configs Reviewers: Matthias J. Sax <matthias@confluent.io>

github_issues:

github_issues_comments:

github_pulls:

1. **title:** KAFKA-7023: Add unit test
   body: Add a unit test that validates after restoreStart, the options are set with bulk loading configs; and after restoreEnd, it resumes to the customized configs ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)

github_pulls_comments:

1. @Ishiihara @mjsax

github_pulls_reviews:

1. As you rewrite this test already: should we use `try-fail-catch` pattern instead of `expected` annotation?
2. Ack, will address before merging.

jira_issues:

1. **summary:** Kafka Streams RocksDB bulk loading config may not be honored with customized RocksDBConfigSetter
   description: We observed frequent L0 -> L1 compaction during Kafka Streams state recovery. Some sample log: {code:java} 2018/06/08-00:04:50.892331 7f8a6d7fa700 (Original Log Time 2018/06/08-00:04:50.892298) [db/compaction_picker_universal.cc:270] [default] Universal: sorted runs files(6): files[3 0 0 0 1 1 38] max score 1.00 2018/06/08-00:04:50.892336 7f8a6d7fa700 (Original Log Time 2018/06/08-00:04:50.892300) [db/compaction_picker_universal.cc:655] [default] Universal: First candidate file 134[0] to reduce size amp. 2018/06/08-00:04:50.892338 7f8a6d7fa700 (Original Log Time 2018/06/08-00:04:50.892302) [db/compaction_picker_universal.cc:686] [default] Universal: size amp not needed. newer-files-total-size 13023497 earliest-file-size 2541530372 2018/06/08-00:04:50.892339 7f8a6d7fa700 (Original Log Time 2018/06/08-00:04:50.892303) [db/compaction_picker_universal.cc:473] [default] Universal: Possible candidate file 134[0]. 2018/06/08-00:04:50.892341 7f8a6d7fa700 (Original Log Time 2018/06/08-00:04:50.892304) [db/compaction_picker_universal.cc:525] [default] Universal: Skipping file 134[0] with size 1007 (compensated size 1287) 2018/06/08-00:04:50.892343 7f8a6d7fa700 (Original Log Time 2018/06/08-00:04:50.892306) [db/compaction_picker_universal.cc:473] [default] Universal: Possible candidate file 133[1]. 2018/06/08-00:04:50.892344 7f8a6d7fa700 (Original Log Time 2018/06/08-00:04:50.892307) [db/compaction_picker_universal.cc:525] [default] Universal: Skipping file 133[1] with size 4644 (compensated size 16124) 2018/06/08-00:04:50.892346 7f8a6d7fa700 (Original Log Time 2018/06/08-00:04:50.892307) [db/compaction_picker_universal.cc:473] [default] Universal: Possible candidate file 126[2]. 2018/06/08-00:04:50.892348 7f8a6d7fa700 (Original Log Time 2018/06/08-00:04:50.892308) [db/compaction_picker_universal.cc:525] [default] Universal: Skipping file 126[2] with size 319764 (compensated size 319764) 2018/06/08-00:04:50.892349 7f8a6d7fa700 (Original Log Time 2018/06/08-

00:04:50.892309) [db/compaction_picker_universal.cc:473] [default] Universal: Possible candidate level 4[3]. 2018/06/08-00:04:50.892351 7f8a6d7fa700 (Original Log Time 2018/06/08-00:04:50.892310) [db/compaction_picker_universal.cc:525] [default] Universal: Skipping level 4[3] with size 2815574 (compensated size 2815574) 2018/06/08-00:04:50.892352 7f8a6d7fa700 (Original Log Time 2018/06/08-00:04:50.892311) [db/compaction_picker_universal.cc:473] [default] Universal: Possible candidate level 5[4]. 2018/06/08-00:04:50.892357 7f8a6d7fa700 (Original Log Time 2018/06/08-00:04:50.892311) [db/compaction_picker_universal.cc:525] [default] Universal: Skipping level 5[4] with size 9870748 (compensated size 9870748) 2018/06/08-00:04:50.892358 7f8a6d7fa700 (Original Log Time 2018/06/08-00:04:50.892313) [db/compaction_picker_universal.cc:473] [default] Universal: Possible candidate level 6[5]. {code} In customized RocksDBConfigSetter, we set {code:java} level0_file_num_compaction_trigger=6 {code} During bulk loading, the following options are set: [https://github.com/facebook/rocksdb/blob/master/options/options.cc] {code:java} Options* Options::PrepareForBulkLoad() { // never slowdown ingest. level0_file_num_compaction_trigger = (1<<30); level0_slowdown_writes_trigger = (1<<30); level0_stop_writes_trigger = (1<<30); soft_pending_compaction_bytes_limit = 0; hard_pending_compaction_bytes_limit = 0; // no auto compactions please. The application should issue a // manual compaction after all data is loaded into L0. disable_auto_compactions = true; // A manual compaction run should pick all files in L0 in // a single compaction run. max_compaction_bytes = (static_cast<uint64_t>(1) << 60); // It is better to have only 2 levels, otherwise a manual // compaction would compact at every possible level, thereby // increasing the total time needed for compactions. num_levels = 2; // Need to allow more write buffers to allow more parallism // of flushes. max_write_buffer_number = 6; min_write_buffer_number_to_merge = 1; // When compaction is disabled, more parallel flush threads can // help with write throughput. max_background_flushes = 4; // Prevent a memtable flush to automatically promote files // to L1. This is helpful so that all files that are // input to the manual compaction are all at L0. max_background_compactions = 2; // The compaction would create large files in L1. target_file_size_base = 256 * 1024 * 1024; return this; } {code} Especially, those values are set to a very large number to avoid compactions and ensures files are all on L0. {code:java} level0_file_num_compaction_trigger = (1<<30); level0_slowdown_writes_trigger = (1<<30); level0_stop_writes_trigger = (1<<30); {code} However, in RockDBStore.java, openDB code, we first call: options.prepareForBulkLoad() and then use the configs from the customized customized RocksDBConfigSetter. This may overwrite the configs set in prepareBulkLoad call. The fix is to move prepareBulkLoad call after applying configs customized RocksDBConfigSetter.

jira_issues_comments:

1. cc [~guozhang] [~bchen225242].
2. Ishiihara opened a new pull request #5166: KAFKA-7023: Move prepareForBulkLoad() call after customized RocksDBConfigSetter URL: https://github.com/apache/kafka/pull/5166 *Summary options.prepareForBulkLoad() and then use the configs from the customized customized RocksDBConfigSetter. This may overwrite the configs set in prepareBulkLoad call. The fix is to move prepareBulkLoad call after applying configs customized RocksDBConfigSetter. *Summary of testing strategy (including rationale) Unit test, test on dev environment on recovery time. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes) -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
3. Thanks for the report! I think this is indeed a perf regression.
4. guozhangwang closed pull request #5166: KAFKA-7023: Move prepareForBulkLoad() call after customized RocksDBConfigSetter URL: https://github.com/apache/kafka/pull/5166 This is a PR merged from a forked repository. As GitHub hides the original diff on merge, it is displayed below for the sake of provenance: As this is a foreign pull request (from a fork), the diff is supplied below (as it won't show otherwise due to GitHub magic): diff --git a/streams/src/main/java/org/apache/kafka/streams/state/internals/RocksDBStore.java b/streams/src/main/java/org/apache/kafka/streams/state/internals/RocksDBStore.java index cfef035a4fd..6084ecbf1e0 100644 --- a/streams/src/main/java/org/apache/kafka/streams/state/internals/RocksDBStore.java +++ b/streams/src/main/java/org/apache/kafka/streams/state/internals/RocksDBStore.java @@ -130,10 +130,6 @@ public void openDB(final ProcessorContext context) { // (this could be a bug in the RocksDB code

and their devs have been contacted). options.setIncreaseParallelism(Math.max(Runtime.getRuntime().availableProcessors(), 2)); - if (prepareForBulkload) { - options.prepareForBulkLoad(); - } - wOptions = new WriteOptions(); wOptions.setDisableWAL(true); @@ -148,6 +144,11 @@ public void openDB(final ProcessorContext context) { final RocksDBConfigSetter configSetter = Utils.newInstance(configSetterClass); configSetter.setConfig(name, options, configs); } + + if (prepareForBulkload) { + options.prepareForBulkLoad(); + } + this.dbDir = new File(new File(context.stateDir(), parentDir), this.name); try { -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

5. guozhangwang opened a new pull request #5197: KAFKA-7023: Add unit test URL: https://github.com/apache/kafka/pull/5197 Add a unit test that validates after restoreStart, the options are set with bulk loading configs; and after restoreEnd, it resumes to the customized configs ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes) -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
6. guozhangwang closed pull request #5197: KAFKA-7023: Add unit test URL: https://github.com/apache/kafka/pull/5197 This is a PR merged from a forked repository. As GitHub hides the original diff on merge, it is displayed below for the sake of provenance: As this is a foreign pull request (from a fork), the diff is supplied below (as it won't show otherwise due to GitHub magic): diff --git a/streams/src/main/java/org/apache/kafka/streams/state/internals/RocksDBStore.java b/streams/src/main/java/org/apache/kafka/streams/state/internals/RocksDBStore.java index 6084ecbf1e0..e858ac07921 100644 --- a/streams/src/main/java/org/apache/kafka/streams/state/internals/RocksDBStore.java +++ b/streams/src/main/java/org/apache/kafka/streams/state/internals/RocksDBStore.java @@ -572,4 +572,9 @@ public void onRestoreEnd(final TopicPartition topicPartition, rocksDBStore.toggleDbForBulkLoading(false); } } + + // for testing + public Options getOptions() { + return options; + } } diff --git a/streams/src/test/java/org/apache/kafka/streams/state/internals/RocksDBStoreTest.java b/streams/src/test/java/org/apache/kafka/streams/state/internals/RocksDBStoreTest.java index b7a9d375d9c..63d877af5d6 100644 --- a/streams/src/test/java/org/apache/kafka/streams/state/internals/RocksDBStoreTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/state/internals/RocksDBStoreTest.java @@ -16,23 +16,21 @@ */ package org.apache.kafka.streams.state.internals; -import org.apache.kafka.common.metrics.Metrics; import org.apache.kafka.common.serialization.Deserializer; import org.apache.kafka.common.serialization.Serdes; import org.apache.kafka.common.serialization.Serializer; import org.apache.kafka.common.serialization.StringDeserializer; import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Bytes; -import org.apache.kafka.common.utils.LogContext; import org.apache.kafka.common.utils.Utils; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.errors.ProcessorStateException; -import org.apache.kafka.streams.processor.internals.MockStreamsMetrics; +import org.apache.kafka.streams.processor.StateRestoreListener; import org.apache.kafka.streams.state.KeyValueIterator; import org.apache.kafka.streams.state.RocksDBConfigSetter; import org.apache.kafka.test.InternalMockProcessorContext; -import org.apache.kafka.test.NoOpRecordCollector; +import org.apache.kafka.test.StreamsTestUtils; import org.apache.kafka.test.TestUtils; import org.junit.After; import org.junit.Before; @@ -43,10 +41,10 @@ import java.io.IOException; import java.io.UnsupportedEncodingException; import java.util.ArrayList; -import java.util.HashMap; import java.util.HashSet; import java.util.List; import java.util.Map; +import java.util.Properties; import java.util.Set; import static org.hamcrest.CoreMatchers.equalTo; @@ -57,8 +55,6 @@ import static org.junit.Assert.fail; public class RocksDBStoreTest { - private final File tempDir = TestUtils.tempDirectory(); - private Serializer<String> stringSerializer = new StringSerializer(); private Deserializer<String> stringDeserializer = new StringDeserializer(); private RocksDBStore rocksDBStore; @@ -67,13 +63,14 @@ @Before public void setUp() { + final Properties

props = StreamsTestUtils.minimalStreamsConfig(); + props.put(StreamsConfig.ROCKSDB_CONFIG_SETTER_CLASS_CONFIG, MockRocksDbConfigSetter.class); rocksDBStore = new RocksDBStore("test"); dir = TestUtils.tempDirectory(); context = new InternalMockProcessorContext(dir, Serdes.String(), Serdes.String(), - new NoOpRecordCollector(), - new ThreadCache(new LogContext("testCache "), 0, new MockStreamsMetrics(new Metrics()))); + new StreamsConfig(props)); } @After @@ -81,6 +78,21 @@ public void tearDown() { rocksDBStore.close(); } + @Test + public void shouldRespectBulkloadOptionsDuringInit() { + rocksDBStore.init(context, rocksDBStore); + + StateRestoreListener restoreListener = context.getRestoreListener(rocksDBStore.name()); + + restoreListener.onRestoreStart(null, rocksDBStore.name(), 0L, 0L); + + assertThat(rocksDBStore.getOptions().level0FileNumCompactionTrigger(), equalTo(1 << 30)); + + restoreListener.onRestoreEnd(null, rocksDBStore.name(), 0L); + + assertThat(rocksDBStore.getOptions().level0FileNumCompactionTrigger(), equalTo(10)); + } + @Test public void shouldNotThrowExceptionOnRestoreWhenThereIsPreExistingRocksDbFiles() throws Exception { rocksDBStore.init(context, rocksDBStore); @@ -108,28 +120,27 @@ public void shouldNotThrowExceptionOnRestoreWhenThereIsPreExistingRocksDbFiles() } @Test - public void verifyRocksDbConfigSetterIsCalled() { - final Map<String, Object> configs = new HashMap<>(); - configs.put(StreamsConfig.APPLICATION_ID_CONFIG, "test-application"); - configs.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "test-server:9092"); - configs.put(StreamsConfig.ROCKSDB_CONFIG_SETTER_CLASS_CONFIG, MockRocksDbConfigSetter.class); + public void shouldCallRocksDbConfigSetter() { MockRocksDbConfigSetter.called = false; - rocksDBStore.openDB(new InternalMockProcessorContext(tempDir, new StreamsConfig(configs))); + + rocksDBStore.openDB(context); assertTrue(MockRocksDbConfigSetter.called); } - @Test(expected = ProcessorStateException.class) - public void shouldThrowProcessorStateExceptionOnOpeningReadOnlyDir() throws IOException { + @Test + public void shouldThrowProcessorStateExceptionOnOpeningReadOnlyDir() { final File tmpDir = TestUtils.tempDirectory(); - InternalMockProcessorContext tmpContext = new InternalMockProcessorContext(tmpDir, - Serdes.String(), - Serdes.Long(), - new NoOpRecordCollector(), - new ThreadCache(new LogContext("testCache "), 0, new MockStreamsMetrics(new Metrics()))); - tmpDir.setReadOnly(); + InternalMockProcessorContext tmpContext = new InternalMockProcessorContext(tmpDir, new StreamsConfig(StreamsTestUtils.minimalStreamsConfig())); + + assertTrue(tmpDir.setReadOnly()); - rocksDBStore.openDB(tmpContext); + try { + rocksDBStore.openDB(tmpContext); + fail("Should have thrown ProcessorStateException"); + } catch (ProcessorStateException e) { + // this is good, do nothing + } } @Test @@ -221,7 +232,7 @@ public void shouldRestoreAll() throws Exception { } @Test - public void shouldPutOnlyIfAbsentValue() throws Exception { + public void shouldPutOnlyIfAbsentValue() { rocksDBStore.init(context, rocksDBStore); final Bytes keyBytes = new Bytes(stringSerializer.serialize(null, "one")); final byte[] valueBytes = stringSerializer.serialize(null, "A"); @@ -237,7 +248,7 @@ public void shouldPutOnlyIfAbsentValue() throws Exception { @Test public void shouldHandleDeletesOnRestoreAll() throws Exception { final List<KeyValue<byte[], byte[]>> entries = getKeyValueEntries(); - entries.add(new KeyValue<>("1".getBytes("UTF-8"), (byte[]) null)); + entries.add(new KeyValue<>("1".getBytes("UTF-8"), null)); rocksDBStore.init(context, rocksDBStore); context.restore(rocksDBStore.name(), entries); @@ -258,7 +269,7 @@ public void shouldHandleDeletesAndPutbackOnRestoreAll() throws Exception { entries.add(new KeyValue<>("1".getBytes("UTF-8"), "a".getBytes("UTF-8"))); entries.add(new KeyValue<>("2".getBytes("UTF-8"), "b".getBytes("UTF-8"))); // this will be deleted - entries.add(new KeyValue<>("1".getBytes("UTF-8"), (byte[]) null)); + entries.add(new KeyValue<>("1".getBytes("UTF-8"), null)); entries.add(new KeyValue<>("3".getBytes("UTF-8"), "c".getBytes("UTF-8"))); // this will restore key "1" as WriteBatch applies updates in order entries.add(new KeyValue<>("1".getBytes("UTF-8"), "restored".getBytes("UTF-8"))); @@ -320,7 +331,7 @@ public void shouldRestoreThenDeleteOnRestoreAll() throws Exception { entries.add(new KeyValue<>("2".getBytes("UTF-8"), "b".getBytes("UTF-8"))); entries.add(new KeyValue<>("3".getBytes("UTF-8"), "c".getBytes("UTF-8"))); - entries.add(new KeyValue<>("1".getBytes("UTF-8"), (byte[]) null)); + entries.add(new KeyValue<>("1".getBytes("UTF-8"), null)); context.restore(rocksDBStore.name(), entries); @@ -342,7 +353,9 @@ public void shouldThrowNullPointerExceptionOnNullPut() { try { rocksDBStore.put(null, stringSerializer.serialize(null, "someVal")); fail("Should have thrown NullPointerException on null put()"); - } catch (NullPointerException e) { } + } catch (NullPointerException e) { + // this is good + } } @Test @@ -351,7 +364,9 @@ public void shouldThrowNullPointerExceptionOnNullPutAll() { try {

rocksDBStore.put(null, stringSerializer.serialize(null, "someVal")); fail("Should have thrown NullPointerException on null put()"); - } catch (NullPointerException e) { } + } catch (NullPointerException e) { + // this is good + } } @Test @@ -360,7 +375,9 @@ public void shouldThrowNullPointerExceptionOnNullGet() { try { rocksDBStore.get(null); fail("Should have thrown NullPointerException on null get()"); - } catch (NullPointerException e) { } + } catch (NullPointerException e) { + // this is good + } } @Test @@ -369,7 +386,9 @@ public void shouldThrowNullPointerExceptionOnDelete() { try { rocksDBStore.delete(null); fail("Should have thrown NullPointerException on deleting null key"); - } catch (NullPointerException e) { } + } catch (NullPointerException e) { + // this is good + } } @Test @@ -378,7 +397,9 @@ public void shouldThrowNullPointerExceptionOnRange() { try { rocksDBStore.range(null, new Bytes(stringSerializer.serialize(null, "2"))); fail("Should have thrown NullPointerException on deleting null key"); - } catch (NullPointerException e) { } + } catch (NullPointerException e) { + // this is good + } } @Test(expected = ProcessorStateException.class) @@ -397,6 +418,8 @@ public void shouldThrowProcessorStateExceptionOnPutDeletedDir() throws IOExcepti @Override public void setConfig(final String storeName, final Options options, final Map<String, Object> configs) { called = true; + + options.setLevel0FileNumCompactionTrigger(10); } } diff --git a/streams/src/test/java/org/apache/kafka/test/InternalMockProcessorContext.java b/streams/src/test/java/org/apache/kafka/test/InternalMockProcessorContext.java index e5571eb43c3..bb42d1c4a26 100644 --- a/streams/src/test/java/org/apache/kafka/test/InternalMockProcessorContext.java +++ b/streams/src/test/java/org/apache/kafka/test/InternalMockProcessorContext.java @@ -78,6 +78,13 @@ public InternalMockProcessorContext(final File stateDir, this(stateDir, null, null, new StreamsMetricsImpl(new Metrics(), "mock"), config, null, null); } + public InternalMockProcessorContext(final File stateDir, + final Serde<?> keySerde, + final Serde<?> valSerde, + final StreamsConfig config) { + this(stateDir, keySerde, valSerde, new StreamsMetricsImpl(new Metrics(), "mock"), config, null, null); + } + public InternalMockProcessorContext(final StateSerdes<?, ?> serdes, final RecordCollector collector) { this(null, serdes.keySerde(), serdes.valueSerde(), collector, null); @@ -293,10 +300,14 @@ public Headers headers() { return Collections.unmodifiableMap(storeMap); } - public void restore(final String storeName, final Iterable<KeyValue<byte[], byte[]>> changeLog) { + public StateRestoreListener getRestoreListener(final String storeName) { + final BatchingStateRestoreCallback restoreCallback = getBatchingRestoreCallback(restoreFuncs.get(storeName)); + return getStateRestoreListener(restoreCallback); + } + public void restore(final String storeName, final Iterable<KeyValue<byte[], byte[]>> changeLog) { final BatchingStateRestoreCallback restoreCallback = getBatchingRestoreCallback(restoreFuncs.get(storeName)); - final StateRestoreListener restoreListener = getStateRestoreListener(restoreCallback); + final StateRestoreListener restoreListener = getRestoreListener(storeName); restoreListener.onRestoreStart(null, storeName, 0L, 0L); --