Item 229
git_comments:

git_commits:

1. **summary:** KAFKA-6568; Log cleaner should check partition state before removal from inProgress map (#4580)
   message: KAFKA-6568; Log cleaner should check partition state before removal from inProgress map (#4580) The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang indefinitely.
   label: code-design

github_issues:

github_issues_comments:

github_pulls:

1. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
   body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
2. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
   body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
   label: code-design
3. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
   body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
4. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
   body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
5. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
   body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
6. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
   body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
7. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
   body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling

`LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)

8. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
   body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
   label: code-design
9. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
   body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
   label: code-design
10. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
    body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
11. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
    body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
12. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
    body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
13. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
    body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
    label: test
14. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
    body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
15. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
    body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
16. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
    body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist

(excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
label: code-design
17. **title:** KAFKA-6568; The log cleaner should check the partition state before r…
body: …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### Committer Checklist (excluded from commit message) - [ ] Verify design and implementation - [ ] Verify test coverage and CI build status - [ ] Verify documentation (including upgrade notes)
label: code-design

github_pulls_comments:

1. ping @ijuma @hachikuji
2. **body:** @hachikuji Thanks for the review. Yes, I agree it would be cleaner to have a separate set to keep the paused partitions.
label: code-design
3. @hachikuji Do you think we should backport this patch to 1.0 and 0.11? 0.10.1 and above are also impacted, but they might be a little too old. That said, our policy is 2 year's support, so maybe we still want to fix versions prior to 0.11.
4. @becketqin I think 1.0 and 0.11 are enough.
5. @ijuma Sounds good.
6. @becketqin I picked back to 0.11.0. Probably worth double-checking the commits to 0.11.0 and 1.0 since there were some trivial conflicts to resolve.
7. @hachikuji Thanks for merging and backporting. The commits look good to me.

github_pulls_reviews:

1. **body:** nit: remove space after `tp`. Same in `setCleaningState`.
label: code-design
2. Maybe we may as well assert the offset was checkpointed as well?
3. The other exceptional case is if `inProgress` does not contain the partition.
4. the inProgress should always contain the partition. Otherwise something is wrong and it is an IllegalStateException.
5. **body:** Right. I'm saying we can add a test for this case.
label: test
6. Ah, got it. Added the test.
7. You can replace 5 lines with something like: ```scala intercept[IllegalStateException] (cleanerManager.doneDeleting(tp)) ``` Similar for the other 2 places.
8. **body:** It would be a bit confusing if `s` is `None`. Maybe we'd want to have an explicit message for that case. Same for the other method.
label: code-design

jira_issues:

1. **summary:** LogCleanerManager.doneDeleting() should check the partition state before deleting the in progress partition
description: {{LogCleanerManager.doneDeleting()}} removes the partition from the {{inProgress}} map without checking if the partition is paused or not. This will cause the paused partition state to be lost, and may also cause another thread calling {{LogCleanerManager.abortAndPauseCleaning()}} to block indefinitely waiting on the partition state to become paused.
label: code-design
2. **summary:** LogCleanerManager.doneDeleting() should check the partition state before deleting the in progress partition
description: {{LogCleanerManager.doneDeleting()}} removes the partition from the {{inProgress}} map without checking if the partition is paused or not. This will cause the paused partition state to be lost, and may also cause another thread calling

{{LogCleanerManager.abortAndPauseCleaning()}} to block indefinitely waiting on the partition state to become paused.

jira_issues_comments:

1. becketqin opened a new pull request #4580: KAFKA-6568; The log cleaner should check the partition state before r… URL: https://github.com/apache/kafka/pull/4580 …emoving it from the inProgress map. The log cleaner should not naively remove the partition from in progress map without checking the partition state. This may cause the other thread calling `LogCleanerManager.abortAndPauseCleaning()` to hang in definitely. ### 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
2. hachikuji closed pull request #4580: KAFKA-6568; The log cleaner should check the partition state before r… URL: https://github.com/apache/kafka/pull/4580 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/core/src/main/scala/kafka/log/LogCleanerManager.scala b/core/src/main/scala/kafka/log/LogCleanerManager.scala index c3d3892aef9..b23107be491 100755 --- a/core/src/main/scala/kafka/log/LogCleanerManager.scala +++ b/core/src/main/scala/kafka/log/LogCleanerManager.scala @@ -96,6 +96,23 @@ private[log] class LogCleanerManager(val logDirs: Seq[File], } } + /** + * Package private for unit test. Get the cleaning state of the partition. + */ + private[log] def cleaningState(tp: TopicPartition): Option[LogCleaningState] = { + inLock(lock) { + inProgress.get(tp) + } + } + + /** + * Package private for unit test. Set the cleaning state of the partition. + */ + private[log] def setCleaningState(tp: TopicPartition, state: LogCleaningState): Unit = { + inLock(lock) { + inProgress.put(tp, state) + } + } /** * Choose the log to clean next and add it to the in-progress set. We recompute this @@ -290,11 +307,11 @@ private[log] class LogCleanerManager(val logDirs: Seq[File], */ def doneCleaning(topicPartition: TopicPartition, dataDir: File, endOffset: Long) { inLock(lock) { - inProgress(topicPartition) match { - case LogCleaningInProgress => + inProgress.get(topicPartition) match { + case Some(LogCleaningInProgress) => updateCheckpoints(dataDir, Option(topicPartition, endOffset)) inProgress.remove(topicPartition) - case LogCleaningAborted => + case Some(LogCleaningAborted) => inProgress.put(topicPartition, LogCleaningPaused) pausedCleaningCond.signalAll() case s => @@ -305,7 +322,15 @@ private[log] class LogCleanerManager(val logDirs: Seq[File], def doneDeleting(topicPartition: TopicPartition): Unit = { inLock(lock) { - inProgress.remove(topicPartition) + inProgress.get(topicPartition) match { + case Some(LogCleaningInProgress) => + inProgress.remove(topicPartition) + case Some(LogCleaningAborted) => + inProgress.put(topicPartition, LogCleaningPaused) + pausedCleaningCond.signalAll() + case s => + throw new IllegalStateException(s"In-progress partition $topicPartition cannot be in $s state.") + } } } @@ -344,7 +369,7 @@ private[log] object LogCleanerManager extends Logging { offset } } - val compactionLagMs = math.max(log.config.compactionLagMs, 0L) // find first segment that cannot be cleaned diff --git a/core/src/test/scala/unit/kafka/log/LogCleanerManagerTest.scala b/core/src/test/scala/unit/kafka/log/LogCleanerManagerTest.scala index 114602919ea..42a447a2b16 100644 --- a/core/src/test/scala/unit/kafka/log/LogCleanerManagerTest.scala +++ b/core/src/test/scala/unit/kafka/log/LogCleanerManagerTest.scala @@ -215,6 +215,76 @@ class LogCleanerManagerTest extends JUnitSuite with Logging { assertEquals(4L, cleanableOffsets._2) } + @Test + def testDoneCleaning(): Unit = { + val logProps = new Properties() + logProps.put(LogConfig.SegmentBytesProp, 1024: java.lang.Integer) + val log = makeLog(config = LogConfig.fromProps(logConfig.originals, logProps)) + while(log.numberOfSegments < 8) + log.appendAsLeader(records(log.logEndOffset.toInt, log.logEndOffset.toInt, time.milliseconds()), leaderEpoch = 0) + + val cleanerManager: LogCleanerManager = createCleanerManager(log) + val tp = new TopicPartition("log", 0) + try { + cleanerManager.doneCleaning(tp, log.dir, 1) + } catch { + case _ : IllegalStateException => + case _ : Throwable => fail("Should have thrown IllegalStateException.") + } + try { + cleanerManager.setCleaningState(tp, LogCleaningPaused) + cleanerManager.doneCleaning(tp, log.dir, 1) + } catch { + case _ : IllegalStateException => + case _ : Throwable => fail("Should have thrown IllegalStateException.") + } + cleanerManager.setCleaningState(tp, LogCleaningInProgress) + cleanerManager.doneCleaning(tp, log.dir, 1) + assertTrue(cleanerManager.cleaningState(tp).isEmpty) +

assertTrue(cleanerManager.allCleanerCheckpoints.get(tp).nonEmpty) + + cleanerManager.setCleaningState(tp, LogCleaningAborted) + cleanerManager.doneCleaning(tp, log.dir, 1) + assertEquals(LogCleaningPaused, cleanerManager.cleaningState(tp).get) + assertTrue(cleanerManager.allCleanerCheckpoints.get(tp).nonEmpty) + } + + @Test + def testDoneDeleting(): Unit = { + val records = TestUtils.singletonRecords("test".getBytes, key="test".getBytes) + val log: Log = createLog(records.sizeInBytes * 5, LogConfig.Compact + "," + LogConfig.Delete) + val cleanerManager: LogCleanerManager = createCleanerManager(log) + + val tp = new TopicPartition("log", 0) + + try { + cleanerManager.doneDeleting(tp) + } catch { + case _ : IllegalStateException => + case _ : Throwable => fail("Should have thrown IllegalStateException.") + } + + try { + cleanerManager.setCleaningState(tp, LogCleaningPaused) + cleanerManager.doneDeleting(tp) + } catch { + case _ : IllegalStateException => + case _ : Throwable => fail("Should have thrown IllegalStateException.") + } + + cleanerManager.setCleaningState(tp, LogCleaningInProgress) + cleanerManager.doneDeleting(tp) + assertTrue(cleanerManager.cleaningState(tp).isEmpty) + + cleanerManager.setCleaningState(tp, LogCleaningAborted) + cleanerManager.doneDeleting(tp) + assertEquals(LogCleaningPaused, cleanerManager.cleaningState(tp).get) + + } + private def createCleanerManager(log: Log): LogCleanerManager = { val logs = new Pool[TopicPartition, Log]() logs.put(new TopicPartition("log", 0), log) ---------- 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. becketqin opened a new pull request #4592: MINOR follow-up for KAFKA-6568 URL: https://github.com/apache/kafka/pull/4592 @ijuma This is the minor follow-up patch for #4580 to address your comments. ### 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
4. ijuma closed pull request #4592: MINOR follow-up for KAFKA-6568 URL: https://github.com/apache/kafka/pull/4592 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/core/src/main/scala/kafka/log/LogCleanerManager.scala b/core/src/main/scala/kafka/log/LogCleanerManager.scala index b23107be491..223c6119654 100755 --- a/core/src/main/scala/kafka/log/LogCleanerManager.scala +++ b/core/src/main/scala/kafka/log/LogCleanerManager.scala @@ -314,6 +314,8 @@ private[log] class LogCleanerManager(val logDirs: Seq[File], case Some(LogCleaningAborted) => inProgress.put(topicPartition, LogCleaningPaused) pausedCleaningCond.signalAll() + case None => + throw new IllegalStateException(s"State for partition $topicPartition should exist.") case s => throw new IllegalStateException(s"In-progress partition $topicPartition cannot be in $s state.") } @@ -328,6 +330,8 @@ private[log] class LogCleanerManager(val logDirs: Seq[File], case Some(LogCleaningAborted) => inProgress.put(topicPartition, LogCleaningPaused) pausedCleaningCond.signalAll() + case None => + throw new IllegalStateException(s"State for partition $topicPartition should exist.") case s => throw new IllegalStateException(s"In-progress partition $topicPartition cannot be in $s state.") } diff --git a/core/src/test/scala/unit/kafka/log/LogCleanerManagerTest.scala b/core/src/test/scala/unit/kafka/log/LogCleanerManagerTest.scala index 42a447a2b16..7455763f5b7 100644 --- a/core/src/test/scala/unit/kafka/log/LogCleanerManagerTest.scala +++ b/core/src/test/scala/unit/kafka/log/LogCleanerManagerTest.scala @@ -226,20 +226,10 @@ class LogCleanerManagerTest extends JUnitSuite with Logging { val cleanerManager: LogCleanerManager = createCleanerManager(log) val tp = new TopicPartition("log", 0) - try { - cleanerManager.doneCleaning(tp, log.dir, 1) - } catch { - case _ : IllegalStateException => - case _ : Throwable => fail("Should have thrown IllegalStateException.") - } - try { - cleanerManager.setCleaningState(tp, LogCleaningPaused) - cleanerManager.doneCleaning(tp, log.dir, 1) - } catch { - case _ : IllegalStateException => - case _ : Throwable => fail("Should have thrown IllegalStateException.") - } + intercept[IllegalStateException](cleanerManager.doneCleaning(tp, log.dir, 1)) + + cleanerManager.setCleaningState(tp, LogCleaningPaused) + intercept[IllegalStateException](cleanerManager.doneCleaning(tp, log.dir, 1)) cleanerManager.setCleaningState(tp, LogCleaningInProgress) cleanerManager.doneCleaning(tp, log.dir, 1) @@ -260,20 +250,10 @@ class LogCleanerManagerTest extends JUnitSuite with Logging { val tp = new TopicPartition("log", 0) - try {

cleanerManager.doneDeleting(tp) - } catch { - case _ : IllegalStateException => - case _ : Throwable => fail("Should have thrown IllegalStateException.") - } - - try { - cleanerManager.setCleaningState(tp, LogCleaningPaused) - cleanerManager.doneDeleting(tp) - } catch { - case _ : IllegalStateException => - case _ : Throwable => fail("Should have thrown IllegalStateException.") - } + intercept[IllegalStateException](cleanerManager.doneDeleting(tp)) + + cleanerManager.setCleaningState(tp, LogCleaningPaused) + intercept[IllegalStateException] (cleanerManager.doneDeleting(tp)) cleanerManager.setCleaningState(tp, LogCleaningInProgress) cleanerManager.doneDeleting(tp) ---------- 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