Item 137
git_comments:

git_commits:

1. **summary:** [SPARK-26267][SS] Retry when detecting incorrect offsets from Kafka (2.4)
   message: [SPARK-26267][SS] Retry when detecting incorrect offsets from Kafka (2.4) ## What changes were proposed in this pull request? Backport #23324 to branch-2.4. ## How was this patch tested? Jenkins Closes #23365 from zsxwing/SPARK-26267-2.4. Authored-by: Shixiong Zhu <zsxwing@gmail.com> Signed-off-by: Shixiong Zhu <zsxwing@gmail.com>

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-26267][SS]Retry when detecting incorrect offsets from Kafka
   body: ## What changes were proposed in this pull request? Due to [KAFKA-7703](https://issues.apache.org/jira/browse/KAFKA-7703), Kafka may return an earliest offset when we are request a latest offset. This will cause Spark to reprocess data. As per suggestion in KAFKA-7703, we put a position call between poll and seekToEnd to block the fetch request triggered by `poll` before calling `seekToEnd`. In addition, to avoid other unknown issues, we also use the previous known offsets to audit the latest offsets returned by Kafka. If we find some incorrect offsets (a latest offset is less than an offset in `knownOffsets`), we will retry at most `maxOffsetFetchAttempts` times. ## How was this patch tested? Jenkins

github_pulls_comments:

1. **[Test build #100169 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/100169/testReport)** for PR 23324 at commit [`ba02dd3`](https://github.com/apache/spark/commit/ba02dd350dad423b30dfef21fff9f7a520f49788). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
2. **[Test build #100255 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/100255/testReport)** for PR 23324 at commit [`e686f10`](https://github.com/apache/spark/commit/e686f10d21294848001b8b4cee02bfb22f09a5a4). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
3. **[Test build #100266 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/100266/testReport)** for PR 23324 at commit [`063d629`](https://github.com/apache/spark/commit/063d6290f88a8adecfd7141aab7d1b99c760d2e3). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
4. **[Test build #100267 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/100267/testReport)** for PR 23324 at commit [`fb1f698`](https://github.com/apache/spark/commit/fb1f698395d16e8a2aabe7441d628526160eca46). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
5. LGTM.
6. Thanks. Merging to master and ~~2.4~~. There are some minor conflicts. Will open a new PR to backport the fix.
7. when can we expect this fix to be in maven ? currently the latest version still has this bug. which I assume related to the error we are seeing? : ``` org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 8.0 failed 4 times, most recent failure: Lost task 0.3 in stage 8.0 (TID 164, qa2-hdp-3.acuityads.org, executor 2): java.lang.AssertionError: assertion failed: latest offset -9223372036854775808 does not equal -1 ```
8. @linehrr this sounds like a different issue. Could you create a JIRA ticket to post details?
9. @zsxwing not too sure if this is actually a bug related to this fix, but I can share my stacktrace here: ``` 19/01/07 22:56:15 ERROR streaming.MicroBatchExecution: Query [id = c46c67ee-3514-4788-8370-a696837b21b1, runId = bb52783e-33d6-460b-9aa2-cc5da414531e] terminated with error org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 8.0 failed 4 times, most recent failure: Lost task 0.3 in stage 8.0 (TID 164, qa2-hdp-3.acuityads.org, executor 2): java.lang.AssertionError: assertion f ailed: latest offset -9223372036854775808 does not equal -1 at scala.Predef$.assert(Predef.scala:170) at org.apache.spark.sql.kafka010.KafkaMicroBatchInputPartitionReader.resolveRange(KafkaMicroBatchReader.scala:371) at org.apache.spark.sql.kafka010.KafkaMicroBatchInputPartitionReader.<init>(KafkaMicroBatchReader.scala:329) at org.apache.spark.sql.kafka010.KafkaMicroBatchInputPartition.createPartitionReader(KafkaMicroBatchReader.scala:314) at org.apache.spark.sql.execution.datasources.v2.DataSourceRDD.compute(DataSourceRDD.scala:42) at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:324) at org.apache.spark.rdd.RDD.iterator(RDD.scala:288) at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52) at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:324) at org.apache.spark.rdd.RDD.iterator(RDD.scala:288) at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52) at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:324) at org.apache.spark.rdd.RDD.iterator(RDD.scala:288) at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52) at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:324) at org.apache.spark.rdd.RDD.iterator(RDD.scala:288) at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:99) at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:55) at org.apache.spark.scheduler.Task.run(Task.scala:121) at org.apache.spark.executor.Executor$TaskRunner$$anonfun$10.apply(Executor.scala:402) at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1360) at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:408) at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142) at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617) at java.lang.Thread.run(Thread.java:745) ``` for some reason, looks like fetchLatestOffset returned a Long.MIN_VALUE for one of the partitions. I checked the structured streaming checkpoint, that was correct, it's the currentAvailableOffset was set to Long.MIN_VALUE. kafka broker version: 1.1.0. lib we used: ```// https://mvnrepository.com/artifact/org.apache.spark/spark-sql-kafka-0-10 libraryDependencies += "org.apache.spark" %% "spark-sql-kafka-0-10" % "2.4.0" ``` how to reproduce: basically we started a structured streamer and subscribed a topic of 4 partitions. then produced some messages into topic, job crashed and logged the stacktrace like above.

10. @linehrr I think this is a different issue. Could you create a JIRA ticket for this? It would be great that you can upload your logs for us to check. Thanks!
11. @zsxwing ok logged here: https://issues.apache.org/jira/browse/KAFKA-7796
12. @linehrr You should file the issue to https://issues.apache.org/jira/projects/SPARK, not KAFKA.
13. @HeartSaVioR I actually can't tell here the bug is from the stacktrace along, since the spark consumer calls Kafka consumer.poll(0) to get the partitionOffsets so it could be either places.
14. @zsxwing found out the issue, it is actually indeed another issue of integer overflow: https://issues.apache.org/jira/browse/SPARK-26718, I will submit a PR to fix this soon.

github_pulls_reviews:

1. This was missing when we wrote Kafka source v2.
2. That's my bad. But no tests caught it :(Is it possible to write a test for this?
3. Add more info why calling this twice works.
4. Well, Kafka returns earliest offset only if there is another call (poll I guess) right before the seekToEnd.
5. super nit: `findIncorrectOffsets` doesnt look like a property so better to call this with `()`
6. Not a big deal at all but I would use string interpolation here. Please ignore this comment if other changes are ready.
7. @tdas Very good call. I found the fix here was not correct when writing a unit test.

jira_issues:

1. **summary:** KafkaConsumer.position may return a wrong offset after "seekToEnd" is called
   description: After "seekToEnd" is called, "KafkaConsumer.position" may return a wrong offset set by another reset request. Here is a reproducer: https://github.com/zsxwing/kafka/commit/4e1aa11bfa99a38ac1e2cb0872c055db56b33246 In this reproducer, "poll(0)" will send an "earliest" request in background. However, after "seekToEnd" is called, due to a race condition in "Fetcher.resetOffsetIfNeeded" (It's not atomic, "seekToEnd" could happen between the check https://github.com/zsxwing/kafka/commit/4e1aa11bfa99a38ac1e2cb0872c055db56b33246#diff-b45245913eaae46aa847d2615d62cde0R585 and the seek https://github.com/zsxwing/kafka/commit/4e1aa11bfa99a38ac1e2cb0872c055db56b33246#diff-b45245913eaae46aa847d2615d62cde0R605), "KafkaConsumer.position" may return an "earliest" offset.

jira_issues_comments:

1. This seems introduced by KAFKA-6397 which moved the offset updating codes to a different thread.
2. I also noticed that TopicPartitionState is modified by multiple threads without any protection.
3. [~zsxwing] I'll pick this up if you don't mind and look into it.
4. [~viktorsomogyi] Cool! Thanks! Let me know if you need more information about the issue.
5. [~zsxwing], I just wanted to send a quick update that I have looked at the code and reproduced it based on your test and now I'm trying to figure out what's the best solution for this. I'll write an update once again when I have some solution proposal.
6. Hi, All. Is there any update for this issue?
7. [~zsxwing], [~dongjoon], I've looked into the issue and it seems there is no easy fix for this in the code as it designed to be async, so it might take some time. Even if we make the method atomic the offset reset that arrives later will be discarded as the first reset nulls out the resetStrategy in SubscriptionState which triggers the {{else if (!subscriptions.isOffsetResetNeeded(partition))}} [check|https://github.com/apache/kafka/blob/trunk/clients/src/main/java/org/apache/kafka/clients/consumer/internals/Fetcher.java#L579] to skip the offset reset. In the current situation as a workaround you could put a {{position}} call between {{poll}} and {{seekToEnd}} as it blocks until {{poll}} returns or some error happens.
8. [~rsivaram], [~hachikuji] I see you modified that part of the code lately, what is your take on this?
9. We stumbled upon this bug in production. [~viktorsomogyi], are you still working on this issue? If not, I can try to continue working on it.
10. Hey [~nikita.glashenko], thanks for bumping this issue. I have a solution just need a day or two to polish it and create a PR. I'll let you know so you and folks on this issue and you can join the code review. Does that work for you?
11. [~viktorsomogyi], yes, thank you.
12. viktorsomogyi commented on pull request #6407: KAFKA-7703: position() may return a wrong offset after seekToEnd URL: https://github.com/apache/kafka/pull/6407 When poll is called which resets the offsets to the beginning, followed by a seekToEnd and a position, it could happen that the "reset to earliest" call in poll overrides the "reset to latest" initiated by seekToEnd in a very delicate way: 1. both request has been issued and returned to the client side (listOffsetResponse has happened) 2. in Fetcher.resetOffsetIfNeeded(TopicPartition, Long, OffsetData) the thread scheduler could prefer the heartbeat thread with the "reset to earliest" call, overriding the offset to the earliest and setting the SubscriptionState with that position. 3. The thread scheduler continues execution of the thread (application thread) with the "reset to latest" call and discards it as the "reset to earliest" already set the position - the wrong one. 4. The blocking position call returns with the earliest offset instead of the latest, despite it wasn't expected. The fix makes the TopicPartitionState in SubscriptionState synchronized and starts to track the requested reset timestamp. With this we can precisely decide if the incoming offset reset is really what we want. Therefore the latest initiated offset reset will happen only. Synchronization furthermore ensures that this is done in an atomic manner to avoid further similar bugs. ### 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 to 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
13. hachikuji commented on pull request #6407: KAFKA-7703: position() may return a wrong offset after seekToEnd URL: https://github.com/apache/kafka/pull/6407 ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on to 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