Item 54
git_comments:

git_commits:

1. **summary:** KAFKA-955 (followup patch) After a leader change, messages sent with ack=0 are lost; reviewed by Neha Narkhede and Jun Rao
   message: KAFKA-955 (followup patch) After a leader change, messages sent with ack=0 are lost; reviewed by Neha Narkhede and Jun Rao

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** After a leader change, messages sent with ack=0 are lost
   description: If the leader changes for a partition, and a producer is sending messages with ack=0, then messages will be lost, since the producer has no active way of knowing that the leader has changed, until it's next metadata refresh update. The broker receiving the message, which is no longer the leader, logs a message like this: Produce request with correlation id 7136261 from client on partition [mytopic,0] failed due to Leader not local for partition [mytopic,0] on broker 508818741 This is exacerbated by the controlled shutdown mechanism, which forces an immediate leader change. A possible solution to this would be for a broker which receives a message, for a topic that it is no longer the leader for (and if the ack level is 0), then the broker could just silently forward the message over to the current leader.

jira_issues_comments:

1. Here's a related scenario, that I see after a rolling restart of my brokers (using ack=0). looking back at my logs, I'm wondering if a producer will reuse the same socket to send data to the same broker, for multiple topics (I'm guessing yes). In which case, it looks like I'm seeing this scenario: 1. producer1 is happily sending messages for topicX and topicY to serverA (serverA is the leader for both topics, only 1 partition for each topic for simplicity). 2. serverA is restarted, and in the process, serverB becomes the new leader for both topicX and topicY. 3. producer1 decides to send a new message to topicX to serverA. 3a. this results in an exception ("Connection reset by peer"). producer1's connection to serverA is invalidated. 3b. producer1 makes a new metadata request for topicX, and learns that serverB is now the leader for topicX. 3c. producer1 resends the message to topicX, on serverB. 4. producer1 decides to send a new message to topicY to serverA. 4a. producer1 notes that it's socket to serverA is invalid, so it creates a new connection to serverA. 4b. producer1 successfully sends it's message to serverA (without realizing that serverA is no longer the leader for topicY). 4c. serverA logs to it's console: 2013-06-23 08:28:46,770 WARN [kafka-request-handler-2] server.KafkaApis - [KafkaApi-508818741] Produce request with correlation id 7136261 from client on partition [mytopic,0] failed due to Leader not local for partition [mytopic,0] on broker 508818741 5. producer1 continues to send messages for topicY to serverA, and serverA continues to log the same messages. 6. 10 minutes later, producer1 decides to update it's metadata for topicY, and learns that serverB is now the leader for topidY. 7. the warning messages finally stop in the console for serverA. I am pretty sure this scenario, or one very close to it, is what I'm seeing in my logs, after doing a rolling restart, with controlled shutdown. I think this scenario makes the issue more severe than just a problem with controlled restart and ack=0.
2. It seems there are various ways that can cause this to happen. (a) In the above scenario, after the leaders fail over, topicX causes new sockets to be established. Then topicY uses the newly established socket without realizing that the leader for topic Y has changed. (b) When we fetch the metadata for a topic, we fetch the metadata for all partitions. Let's say that we never get to send any data to a particular partition. The socket for this partition is not established since SyncProducer make socket connections lazily on first send. Then the leader for the partition changes. Finally, the producer sends a message to that partition. Now a socket is established to the wrong leader without the producer realizing it. In general, if we hit any error for produce requests with ack=0, currently the producer won't notice it. For example, if the broker hits a MessageTooLargeException or if the broker hits any other unexpected exceptions. In those cases, forwarding the requests will not help. Also, forwarding requests will complicate the logic in the broker since we have to figure out the broker's host and port, and potentially cache the socket connection to other brokers. An alternative solution is to simply close the socket connection when we hit any error for produce requests with ack=0. This way, the producer will realize the error on next send.
3. Following the close-socket approach, I propose the following change: 1. Add a closeSocket: Boolean field in Response class. 2. In KafkaApi.handleProducerRequest, if requireAck == 0 check if numPartitionsInError != 0. If yes set closeSocket to true in the returning response. 3. SocketServer.Processor.processNewResponses, if curr.responseSend == null, check if closeSocket == true. If yes, log the closing socket info and close the key.
4. Add one case for ack=0 in testSendWithDeadBroker, passed.
5. Thanks for the patch. Some comments: 1. SocketServer: We should call updateRequestMetrics even when we close the socket. Otherwise, total time will be broken for that request. 2. ProducerTest: Let's add a new unit test instead of piggybacking on the existing one. What we can do is to create a sync producer and send a produce request with ack=0 that will introduce an error (e.g., a message larger than max size). After that, we can verified that the underlying socket is closed. 3. KafkaApi: In the debug logging, why not log the whole producer request?
6. Thanks for the comments Jun. 1,2,3. Done.
7. Thanks for patch v2. Some more comments. 20. testSendWithAckZeroDeadBroker(): I am not sure if the unit test does what you want. First of all, setup() will always start brokers for each test unless you explicitly shut them down. So, in this test, the brokers are not dead. Second, the test doesn't really test that the socket is closed after error. I suggest that we add a new test in SyncProducerTest. We send a request with ack=0 with a large message. After that, we can try to send a new request again and we should hit a socket I/O exception. We may have to wait for some time between the two requests.
8. Sorry for the name misleading, I did not shut down the broker but instead send a large message to it to trigger the MessageSizeTooLargeException. The name of the test should be testSendTooLargeMessageWithAckZero. I will use SyncProducer instead

of Producer in this test, and send a normal message to the broker after this message, and expecting it to fail due to socket IO exception.

9. Add the testMessageSizeTooLargeWithAckZero to syncProducerTest, which: 1. First send a large message that will cause the MessageSizeTooLarge exception, and hence close the socket. But this message will be silently dropped and lost. 2. Then send another large message, but just to make sure its size exceeds the buffer size so the socket buffer will be flushed immediately; this send should fail since the socket has been closed.
10. If the producer is sending messages through the same broker for other topic+partitions that did not have a leader change they will also be affected by the close of the socket, resulting in lost messages. It would be better if the broker would notify all connected clients of broker changes (leader change, broker add/delete, topic add/delete) by sending an unsolicited MetadataResponse message (with corrid 0) (or by some other mean). This would propogate topology changes in a faster and less intrusive way.
11. Hello Magnus, 1. Under Ack=0, the producer does not expect any responses for produce request, and it does not listen to any possible connections from the producer either. So actively sending MetadataResponse would not work: producers are only expecting MetadataResponse when they send MetadataRequest. 2. When we close the socket, producer would be notified and try to refresh their Metadata and retry. Since by default each produce request will be retried multiple times before it is got dropped, the current approach would not cause lost messages.
12. Hi Guozhang, I understand that you might not want to introduce a new message semantic at this point of the 0.8 beta, but it wont get easier after the release. My proposal is a change of the protocol definition to allow unsolicited metadata response messages to be sent from the broker, this would of course require changes in most clients, but a very small one for those that are not interested in keeping their leader cache up to date. Consider a producer forwarding >100kmsgs/s for a number of topics to a broker that suddenly drops the connection because one of those topics changed leader, the producer message queue will quickly build up and might start dropping messages (for topics that didnt loose their leader) due to local queue thresholds or very slowly recover if the current rate of messages is close to the maximum thruput. In my mind closing the socket because one top+par changed leader is a very intrusive way to signal an event for sub-set of the communication, and it should instead be fixed properly with an unsoliticed metadata response message. The unsolicited metadata response message is useful for other scenarios aswell, new brokers and topics being added, for instance. My two cents on the topic, thank you.
13. Magnus, thanks for your comment. What you suggested is interesting and could be a more effective way of communicating between the producer and the broker. It does require that the producer be able to receive requests initiated at the broker. We do plan to make the producer side processing selector based for efficiency reason. However, this will be a post 0.8 item. We could consider your suggestion then. Regarding your concern about dropped messages, my take is the following. If a client chooses not to receive an ack, it probably means that losing a few batch of messages is not that important. If a client does care about data loss, it can choose ack with 1 or -1. The throughout will be less. However, there are other ways to improve the throughput (e.g., using a larger batch size and/or more instances of producers). Guozhang, patch v3 looks good to me overall. A few more comments: 30. SyncProducerTest.testMessagesizeTooLargeWithAckZero(): You hardcoded the sleep to 500ms. Could you change it to the waitUntil style wait such that the test can finish early if the conditions have been met? 31. KafkaApi.handleProducerRequest(): The logging should probably be at debug level since this doesn't indicate an error at the broker. It's really an error for the client.
14. Thanks for the comments Jun. 30. Done. 31. After a second thought I realized that we do not need to sleep since the second message size is large enough to cause the socket buffer to flush immediately, and by then the socket close should have been triggered by the server. This has been verified in the unit test. Made some minor changes on comments and rebased on 0.8
15. Thanks for the patches, Guozhang. I reviewed patch v4 and here are some comments - KafkaApis and SocketServer 1.1 One way to allow the socket server to close the channel is to just mark the request's key cancelled in the Response object. This way when the socket server is handling the response, it will throw a CancelledKeyException and we close the key in this case. One advantage of this approach is we can avoid introducing the close socket flag, just to handle this case. To make sure the request metrics are always updated, we can move curr.request.updateRequestMetrics to the first statement in the (curr.responseSend == null) block. 1.2 I think the below warn message can be improved - Sending the close socket signal due to error handling produce request [%s] with Ack=0 Let's include the client id, correlation id and list of topics and partitions that this request had. This is probably more useful than printing the entire produce request as is, since that attempts to print things like ByteBufferMessageSet and is unreadable.
16. Thanks for the comments Neha. 1.1. Great point. The only concern is that now KafkaApis need to know that requestKey is actually java.nio.channels.SelectionKey. But I think this is fine. 1.2. Done.
17. After talking around with people I now proposed an approach similar to v4 but generalized with a responseCode instead of just a close socket flag. And on SocketServer the processor would act based on the code instead of checking if the responseSend is null or not. Also change aliveBrokers in KafkaApis from var to val since it is not overwritten in lifetime.
18. I like the way the responseCode is generalized. Patch v6 looks good, few minor comments before checkin - 1. Remove unused variable allBrokers from KafkaApis 2. This comment needs to be changed according to the new response code logic - // a null response send object indicates Maybe we should wait for review from [~jkreps] since he has most context on the socket server.
19. Great fix. A few minor comments, mostly stylistic. RequestChannel.scala: 1. This usage exposes a bit much: requestChannel.sendResponse(new RequestChannel.Response(request.processor, request, null, RequestChannel.CloseSocket)) I think it might be nicer to have this instead: requestChannel.close(request.processor, request) and requestChannel.noResponse(req.processor, request) Implementation would be the same, it just would just be a little more clear for the user and the response codes can be private. Likewise in the response object I should be able to 2. These are a little confusing: val SendResponse: Short = 0 val NoResponse: Short = 1 val CloseSocket: Short = 9 Why is it 0, 1, and 9? What is the relationship between these and ErrorMapping? It should be clear from reading. Is there a reason we can't use a case class case class ResponseAction case object SendAction extends ResponseAction case object NoOpAction extends ResponseAction case object CloseConnectionAction extends ResponseAction Then to use it response.action match { case SendAction => do send case NoOpAction => read more case CloseConnectionAction => something } This seems clearer to me and I don't think it is significantly more expensive. Can we also standardize the usage so that we no longer have the user EITHER give null or NoResponse? It should be one or the other. 3. This logging "Cancelling the request key to notify socket server close the connection due to error handling produce request " is not informative to the user. What does it mean to cancel a key? What broke? What should they do? I also think this should be info unless we want the server admin to take some action (I don't think so, right? This is a normal occurance). SocketServer.scala 4. The comment "a null response send object" is retained but we are no longer using null to indicate this we are using RequestChannel.NoResponse. I think this comment is actually a little verbose given that we now have a nicely named response action. ProducerTest.scala: 5. org.scalatest.TestFailedException: Is there a reason you are giving the full path here instead of importing it Question on testing, what is the message loss rate with acks=0 under moderate load if we do something like a controlled shutdown with other replicas available?
20. Thanks for the comments, Neha, Jay. Neha: 1. Done. 2. Incorporated with Jay's comments. Jay: 1. Done. 2. Done. I ended up using trait and objects, and let requestChannel.close and requestChannel.noOperation (I changed the name from noResponse here since it matches the noOpAction better) create new responses themselves. 3. Done. 4. Done. 5. Done. Regarding your question, the message loss is depending on the producer throughput and queue size. Since the first message will always be silently dropped, and once the producer

noticed the socket closure, it will stop producing and refresh new metadata, and if during this time the producer queue is full then it will drop more messages. So the answer would be the range between [1, produce-throughput / time-taken-to-refresh-metadata - queue-size].

21. +1 Gorgeous.
22. This is great. +1. One improvement on logging - info(("Send the close connection response due to error handling produce request " + " [clientId = %s, correlationId = %s, topicAndPartition = %s] with Ack=0") .format(produceRequest.clientId, produceRequest.correlationId, produceRequest.topicPartitionMessageSizeMap.mkString("[",",","]"))) Here we only want to print the topic and partition, so it seems that we should be printing the keys of the map, not the entire map ? produceRequest.topicPartitionMessageSizeMap.keySet.mkString(",") I can make this change on checkin.
23. Committed patch v7 to 0.8 after making the logging fix described above
24. Thanks for patch v7. A couple of more comments. 70. There is a long standing bug in ProducerRequest.handleError(). If ack=0, we shouldn't send a response when the broker hits an unexpected error. We should either close the socket connection or send no response. Not sure which one is better. 71. A minor issue. The following comment in RequestChannel is a bit confusing. It sounds like that it needs to read more data from network to complete this request, but it is not. /** No operation to take for the request, need to read more over the network */ def noOperation(processor: Int, request: RequestChannel.Request) {
25. Created reviewboard https://reviews.apache.org/r/14140/
26. Thanks for the followup patch. +1 and committed to 0.8.
27. Hi All, affected version & fixed version is same, just want to know if this fix is available in "0.8.2" I'm facing similar issue in "0.8.2". Thanks.
28. I am using Flume with Kafka Channel & facing below issues. Kafka Version: kafka_2.9.1-0.8.2.0 Flume Version: apache-flume-1.6.0 It seems was resolved from below : Step 1: copy zookeeper Jar file to Flume classpath Step 2: a1.channels.c1.kafka.producer.type = async Note: i didn't change default value of request.required.acks. It seems works, it is still in testing... ~~~~~~ Issue 1: 02 Nov 2016 22:20:06,201 WARN [ConsumerFetcherThread-flume_tbox-topic_SATL2036-1478087994030-54387da2-0-2] (kafka.utils.Logging$class.warn:83) - Reconnect due to socket error: null 02 Nov 2016 22:20:06,203 INFO [ConsumerFetcherThread-flume_tbox-topic_SATL2036-1478087994030-54387da2-0-2] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume_tbox-topic_SATL2036-1478087994030-54387da2-0-2], Stopped 02 Nov 2016 22:20:06,203 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume_tbox-topic_SATL2036-1478087994030-54387da2-0-2], Shutdown completed 02 Nov 2016 22:20:06,203 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume_tbox-topic_SATL2036-1478087994030-54387da2-0-1], Shutting down 02 Nov 2016 22:20:06,204 WARN [ConsumerFetcherThread-flume_tbox-topic_SATL2036-1478087994030-54387da2-0-1] (kafka.utils.Logging$class.warn:83) - Reconnect due to socket error: null 02 Nov 2016 22:20:06,204 INFO [ConsumerFetcherThread-flume_tbox-topic_SATL2036-1478087994030-54387da2-0-1] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume_tbox-topic_SATL2036-1478087994030-54387da2-0-1], Stopped 02 Nov 2016 22:20:06,204 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume_tbox-topic_SATL2036-1478087994030-54387da2-0-1], Shutdown completed 02 Nov 2016 22:20:06,205 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherManager-1478087994042] All connections stopped 02 Nov 2016 22:20:06,207 INFO [ZkClient-EventThread-58-SATL2036:2181,SATL2037:2181,SATL2038:2181/kafka] (org.I0Itec.zkclient.ZkEventThread.run:82) - Terminate ZkClient event thread. 02 Nov 2016 22:20:06,212 WARN [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.warn:89) - Failed to send producer request with correlation id 34198503 to broker 1 with data for partitions [channel-tbox-parsed-topic,3] java.nio.channels.ClosedByInterruptException at java.nio.channels.spi.AbstractInterruptibleChannel.end(AbstractInterruptibleChannel.java:202) at sun.nio.ch.SocketChannelImpl.write(SocketChannelImpl.java:511) at java.nio.channels.SocketChannel.write(SocketChannel.java:502) at kafka.network.BoundedByteBufferSend.writeTo(BoundedByteBufferSend.scala:56) at kafka.network.Send$class.writeCompletely(Transmission.scala:75) at kafka.network.BoundedByteBufferSend.writeCompletely(BoundedByteBufferSend.scala:26) at kafka.network.BlockingChannel.send(BlockingChannel.scala:92) at kafka.producer.SyncProducer.liftedTree1$1(SyncProducer.scala:72) at kafka.producer.SyncProducer.kafka$producer$SyncProducer$$doSend(SyncProducer.scala:71) at kafka.producer.SyncProducer$$anonfun$send$1$$anonfun$apply$mcVsp1.applymcVsp(SyncProducer.scala:102) at kafka.producer.SyncProducer$$anonfun$send$1$$anonfun$apply$mcVsp1.apply(SyncProducer.scala:102) at kafka.producer.SyncProducer$$anonfun$send$1$$anonfun$apply$mcVsp1.apply(SyncProducer.scala:102) at kafka.metrics.KafkaTimer.time(KafkaTimer.scala:33) at kafka.producer.SyncProducer$$anonfun$send$1.applymcVsp(SyncProducer.scala:101) at kafka.producer.SyncProducer$$anonfun$send$1.apply(SyncProducer.scala:101) at kafka.producer.SyncProducer$$anonfun$send$1.apply(SyncProducer.scala:101) at kafka.metrics.KafkaTimer.time(KafkaTimer.scala:33) at kafka.producer.SyncProducer.send(SyncProducer.scala:100) at kafka.producer.async.DefaultEventHandler.kafka$producer$async$DefaultEventHandler$$send(DefaultEventHandler.scala:255) at kafka.producer.async.DefaultEventHandler$$anonfun$dispatchSerializedData$2.apply(DefaultEventHandler.scala:106) at kafka.producer.async.DefaultEventHandler$$anonfun$dispatchSerializedData$2.apply(DefaultEventHandler.scala:100) at scala.collection.TraversableLike$WithFilter$$anonfun$foreach$1.apply(TraversableLike.scala:772) at scala.collection.mutable.HashMap$$anonfun$foreach$1.apply(HashMap.scala:98) at scala.collection.mutable.HashMap$$anonfun$foreach$1.apply(HashMap.scala:98) at scala.collection.mutable.HashTable$class.foreachEntry(HashTable.scala:226) at scala.collection.mutable.HashMap.foreachEntry(HashMap.scala:39) at scala.collection.mutable.HashMap.foreach(HashMap.scala:98) at scala.collection.TraversableLike$WithFilter.foreach(TraversableLike.scala:771) at kafka.producer.async.DefaultEventHandler.dispatchSerializedData(DefaultEventHandler.scala:100) at kafka.producer.async.DefaultEventHandler.handle(DefaultEventHandler.scala:72) at kafka.producer.Producer.send(Producer.scala:76) at kafka.javaapi.producer.Producer.send(Producer.scala:42) at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doCommit(KafkaChannel.java:357) at org.apache.flume.channel.BasicTransactionSemantics.commit(BasicTransactionSemantics.java:151) at org.apache.flume.channel.ChannelProcessor.processEventBatch(ChannelProcessor.java:192) at org.apache.flume.source.kafka.KafkaSource.process(KafkaSource.java:130) at org.apache.flume.source.PollableSourceRunner$PollingRunner.run(PollableSourceRunner.java:139) at java.lang.Thread.run(Thread.java:745) 02 Nov 2016 22:20:06,214 INFO [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.info:68) - Back off for 100 ms before retrying send. Remaining retries = 3 02 Nov 2016 22:20:06,214 WARN [PollableSourceRunner-KafkaSource-r1] (org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doCommit:363) - Sending events to Kafka failed java.lang.InterruptedException: sleep interrupted at java.lang.Thread.sleep(Native Method) at kafka.producer.async.DefaultEventHandler.handle(DefaultEventHandler.scala:76) at kafka.producer.Producer.send(Producer.scala:76) at

kafka.javaapi.producer.Producer.send(Producer.scala:42) at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doCommit(KafkaChannel.java:357) at
org.apache.flume.channel.BasicTransactionSemantics.commit(BasicTransactionSemantics.java:151) at
org.apache.flume.channel.ChannelProcessor.processEventBatch(ChannelProcessor.java:192) at
org.apache.flume.source.kafka.KafkaSource.process(KafkaSource.java:130) at
org.apache.flume.source.PollableSourceRunner$PollingRunner.run(PollableSourceRunner.java:139) at
java.lang.Thread.run(Thread.java:745) 02 Nov 2016 22:20:06,215 ERROR [PollableSourceRunner-KafkaSource-r1]
(org.apache.flume.source.kafka.KafkaSource.process:153) - KafkaSource EXCEPTION, {} org.apache.flume.ChannelException: Unable
to put batch on required channel: org.apache.flume.channel.kafka.KafkaChannel{name: c1} at
org.apache.flume.channel.ChannelProcessor.processEventBatch(ChannelProcessor.java:200) at
org.apache.flume.source.kafka.KafkaSource.process(KafkaSource.java:130) at
org.apache.flume.source.PollableSourceRunner$PollingRunner.run(PollableSourceRunner.java:139) at
java.lang.Thread.run(Thread.java:745) Caused by: org.apache.flume.ChannelException: Commit failed as send to Kafka failed at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doCommit(KafkaChannel.java:364) at
org.apache.flume.channel.BasicTransactionSemantics.commit(BasicTransactionSemantics.java:151) at
org.apache.flume.channel.ChannelProcessor.processEventBatch(ChannelProcessor.java:192) ... 3 more Caused by:
java.lang.InterruptedException: sleep interrupted at java.lang.Thread.sleep(Native Method) at
kafka.producer.async.DefaultEventHandler.handle(DefaultEventHandler.scala:76) at kafka.producer.Producer.send(Producer.scala:76) at
kafka.javaapi.producer.Producer.send(Producer.scala:42) at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doCommit(KafkaChannel.java:357) ... 5 more 02 Nov 2016
22:20:06,233 INFO [agent-shutdown-hook] (org.apache.zookeeper.ZooKeeper.close:684) - Session: 0x2581dc726ab01ad closed 02 Nov
2016 22:20:06,233 INFO [lifecycleSupervisor-1-1-EventThread] (org.apache.zookeeper.ClientCnxn$EventThread.run:512) - EventThread
shut down 02 Nov 2016 22:20:06,239 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [flume_tbox-topic_SATL2036-
1478087994030-54387da2], ZKConsumerConnector shut down completed 02 Nov 2016 22:20:06,239 INFO [agent-shutdown-hook]
(org.apache.flume.instrumentation.MonitoredCounterGroup.stop:150) - Component type: SOURCE, name: r1 stopped 02 Nov 2016
22:20:06,239 INFO [agent-shutdown-hook] (org.apache.flume.instrumentation.MonitoredCounterGroup.stop:156) - Shutdown Metric for
type: SOURCE, name: r1. source.start.time == 1478087994119 02 Nov 2016 22:20:06,239 INFO [agent-shutdown-hook]
(org.apache.flume.instrumentation.MonitoredCounterGroup.stop:162) - Shutdown Metric for type: SOURCE, name: r1. source.stop.time
== 1478096406239 02 Nov 2016 22:20:06,239 INFO [agent-shutdown-hook]
(org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: SOURCE, name: r1.
source.kafka.commit.time == 555 02 Nov 2016 22:20:06,239 INFO [agent-shutdown-hook]
(org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: SOURCE, name: r1.
source.kafka.event.get.time == 1409513 02 Nov 2016 22:20:06,239 INFO [agent-shutdown-hook]
(org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: SOURCE, name: r1. src.append-
batch.accepted == 0 02 Nov 2016 22:20:06,240 INFO [agent-shutdown-hook]
(org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: SOURCE, name: r1. src.append-
batch.received == 0 02 Nov 2016 22:20:06,240 INFO [agent-shutdown-hook]
(org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: SOURCE, name: r1.
src.append.accepted == 0 02 Nov 2016 22:20:06,240 INFO [agent-shutdown-hook]
(org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: SOURCE, name: r1.
src.append.received == 0 02 Nov 2016 22:20:06,240 INFO [agent-shutdown-hook]
(org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: SOURCE, name: r1.
src.events.accepted == 343 02 Nov 2016 22:20:06,240 INFO [agent-shutdown-hook]
(org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: SOURCE, name: r1.
src.events.received == 343 02 Nov 2016 22:20:06,240 INFO [agent-shutdown-hook]
(org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: SOURCE, name: r1. src.open-
connection.count == 0 02 Nov 2016 22:20:06,240 INFO [agent-shutdown-hook] (org.apache.flume.source.kafka.KafkaSource.stop:237) -
Kafka Source r1 stopped. Metrics: SOURCE:r1{src.events.accepted=343, src.open-connection.count=0, src.append.received=0,
source.kafka.event.get.time=1409513, src.append-batch.received=0, src.append-batch.accepted=0, src.append.accepted=0,
src.events.received=343, source.kafka.commit.time=555} 02 Nov 2016 22:20:06,240 INFO [agent-shutdown-hook]
(kafka.utils.Logging$class.info:68) - [flume-channel-tbox-topic_SATL2036-1478087992871-1a60fb21], ZKConsumerConnector shutting
down 02 Nov 2016 22:20:06,241 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherManager-
1478087993018] Stopping leader finder thread 02 Nov 2016 22:20:06,241 INFO [agent-shutdown-hook]
(kafka.utils.Logging$class.info:68) - [flume-channel-tbox-topic_SATL2036-1478087992871-1a60fb21-leader-finder-thread], Shutting
down 02 Nov 2016 22:20:06,241 INFO [flume-channel-tbox-topic_SATL2036-1478087992871-1a60fb21-leader-finder-thread]
(kafka.utils.Logging$class.info:68) - [flume-channel-tbox-topic_SATL2036-1478087992871-1a60fb21-leader-finder-thread], Stopped 02
Nov 2016 22:20:06,241 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [flume-channel-tbox-topic_SATL2036-
1478087992871-1a60fb21-leader-finder-thread], Shutdown completed 02 Nov 2016 22:20:06,241 INFO [agent-shutdown-hook]
(kafka.utils.Logging$class.info:68) - [ConsumerFetcherManager-1478087993018] Stopping all fetchers 02 Nov 2016 22:20:06,241 INFO
[agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume-channel-tbox-topic_SATL2036-
1478087992871-1a60fb21-0-1], Shutting down 02 Nov 2016 22:20:06,242 WARN [ConsumerFetcherThread-flume-channel-tbox-
topic_SATL2036-1478087992871-1a60fb21-0-1] (kafka.utils.Logging$class.warn:83) - Reconnect due to socket error: null 02 Nov 2016
22:20:06,242 INFO [ConsumerFetcherThread-flume-channel-tbox-topic_SATL2036-1478087992871-1a60fb21-0-1]
(kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume-channel-tbox-topic_SATL2036-1478087992871-1a60fb21-0-1],
Stopped 02 Nov 2016 22:20:06,242 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume-
channel-tbox-topic_SATL2036-1478087992871-1a60fb21-0-1], Shutdown completed 02 Nov 2016 22:20:06,242 INFO [agent-shutdown-
hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherManager-1478087993018] All connections stopped 02 Nov 2016
22:20:06,243 INFO [ZkClient-EventThread-42-SATL2036:2181,SATL2037:2181,SATL2038:2181/kafka]
(org.I0Itec.zkclient.ZkEventThread.run:82) - Terminate ZkClient event thread. 02 Nov 2016 22:20:06,244 INFO [agent-shutdown-hook]
(org.apache.zookeeper.ZooKeeper.close:684) - Session: 0x356eb2d4b833fab closed 02 Nov 2016 22:20:06,244 INFO [agent-shutdown-
hook] (kafka.utils.Logging$class.info:68) - [flume-channel-tbox-topic_SATL2036-1478087992871-1a60fb21], ZKConsumerConnector
shut down completed 02 Nov 2016 22:20:06,244 INFO [SinkRunner-PollingRunner-FailoverSinkProcessor-EventThread]
(org.apache.zookeeper.ClientCnxn$EventThread.run:512) - EventThread shut down 02 Nov 2016 22:20:06,246 INFO [agent-shutdown-
hook] (kafka.utils.Logging$class.info:68) - Shutting down producer 02 Nov 2016 22:20:06,247 INFO [agent-shutdown-hook]
(kafka.utils.Logging$class.info:68) - Closing all sync producers 02 Nov 2016 22:20:06,256 INFO [agent-shutdown-hook]
(kafka.utils.Logging$class.info:68) - Disconnecting from 10.25.20.36:9092 02 Nov 2016 22:20:06,256 INFO [agent-shutdown-hook]

(org.apache.flume.instrumentation.MonitoredCounterGroup.stop:150) - Component type: CHANNEL, name: c1 stopped 02 Nov 2016 22:20:06,256 INFO [agent-shutdown-hook] (org.apache.flume.instrumentation.MonitoredCounterGroup.stop:156) - Shutdown Metric for type: CHANNEL, name: c1. channel.start.time == 1478087992836 02 Nov 2016 22:20:06,256 INFO [agent-shutdown-hook] (org.apache.flume.instrumentation.MonitoredCounterGroup.stop:162) - Shutdown Metric for type: CHANNEL, name: c1. channel.stop.time == 1478096406256 02 Nov 2016 22:20:06,257 INFO [agent-shutdown-hook] (org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: CHANNEL, name: c1. channel.capacity == 0 02 Nov 2016 22:20:06,257 INFO [agent-shutdown-hook] (org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: CHANNEL, name: c1. channel.current.size == 0 02 Nov 2016 22:20:06,257 INFO [agent-shutdown-hook] (org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: CHANNEL, name: c1. channel.event.put.attempt == 0 02 Nov 2016 22:20:06,257 INFO [agent-shutdown-hook] (org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: CHANNEL, name: c1. channel.event.put.success == 343 02 Nov 2016 22:20:06,257 INFO [agent-shutdown-hook] (org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: CHANNEL, name: c1. channel.event.take.attempt == 0 02 Nov 2016 22:20:06,257 INFO [agent-shutdown-hook] (org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: CHANNEL, name: c1. channel.event.take.success == 342 02 Nov 2016 22:20:06,257 INFO [agent-shutdown-hook] (org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: CHANNEL, name: c1. channel.kafka.commit.time == 201 02 Nov 2016 22:20:06,258 INFO [agent-shutdown-hook] (org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: CHANNEL, name: c1. channel.kafka.event.get.time == 531 02 Nov 2016 22:20:06,258 INFO [agent-shutdown-hook] (org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: CHANNEL, name: c1. channel.kafka.event.send.time == 1789 02 Nov 2016 22:20:06,258 INFO [agent-shutdown-hook] (org.apache.flume.instrumentation.MonitoredCounterGroup.stop:178) - Shutdown Metric for type: CHANNEL, name: c1. channel.rollback.count == 0 02 Nov 2016 22:20:06,258 INFO [agent-shutdown-hook] (org.apache.flume.channel.kafka.KafkaChannel.stop:123) - Kafka channel c1 stopped. Metrics: CHANNEL:c1{channel.event.put.attempt=0, channel.event.put.success=343, channel.kafka.event.get.time=531, channel.current.size=0, channel.event.take.attempt=0, channel.event.take.success=342, channel.kafka.event.send.time=1789, channel.capacity=0, channel.kafka.commit.time=201, channel.rollback.count=0} 02 Nov 2016 22:20:06,264 WARN [SinkRunner-PollingRunner-DefaultSinkProcessor] (org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake:332) - Error while getting events from Kafka java.lang.InterruptedException at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.reportInterruptAfterWait(AbstractQueuedSynchronizer.java:2014) at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(AbstractQueuedSynchronizer.java:2088) at java.util.concurrent.LinkedBlockingQueue.poll(LinkedBlockingQueue.java:467) at kafka.consumer.ConsumerIterator.makeNext(ConsumerIterator.scala:65) at kafka.consumer.ConsumerIterator.makeNext(ConsumerIterator.scala:33) at kafka.utils.IteratorTemplate.maybeComputeNext(IteratorTemplate.scala:66) at kafka.utils.IteratorTemplate.hasNext(IteratorTemplate.scala:58) at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:306) at org.apache.flume.channel.BasicTransactionSemantics.take(BasicTransactionSemantics.java:113) at org.apache.flume.channel.BasicChannelSemantics.take(BasicChannelSemantics.java:95) at org.apache.flume.sink.kafka.KafkaSink.process(KafkaSink.java:97) at org.apache.flume.sink.DefaultSinkProcessor.process(DefaultSinkProcessor.java:68) at org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:147) at java.lang.Thread.run(Thread.java:745) 02 Nov 2016 22:20:06,274 ERROR [SinkRunner-PollingRunner-DefaultSinkProcessor] (org.apache.flume.sink.kafka.KafkaSink.process:139) - Failed to publish events org.apache.flume.ChannelException: Error while getting events from Kafka at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:333) at org.apache.flume.channel.BasicTransactionSemantics.take(BasicTransactionSemantics.java:113) at org.apache.flume.channel.BasicChannelSemantics.take(BasicChannelSemantics.java:95) at org.apache.flume.sink.kafka.KafkaSink.process(KafkaSink.java:97) at org.apache.flume.sink.DefaultSinkProcessor.process(DefaultSinkProcessor.java:68) at org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:147) at java.lang.Thread.run(Thread.java:745) Caused by: java.lang.InterruptedException at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.reportInterruptAfterWait(AbstractQueuedSynchronizer.java:2014) at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(AbstractQueuedSynchronizer.java:2088) at java.util.concurrent.LinkedBlockingQueue.poll(LinkedBlockingQueue.java:467) at kafka.consumer.ConsumerIterator.makeNext(ConsumerIterator.scala:65) at kafka.consumer.ConsumerIterator.makeNext(ConsumerIterator.scala:33) at kafka.utils.IteratorTemplate.maybeComputeNext(IteratorTemplate.scala:66) at kafka.utils.IteratorTemplate.hasNext(IteratorTemplate.scala:58) at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:306) ... 6 more 02 Nov 2016 22:20:06,275 ERROR [SinkRunner-PollingRunner-DefaultSinkProcessor] (org.apache.flume.SinkRunner$PollingRunner.run:160) - Unable to deliver event. Exception follows. org.apache.flume.EventDeliveryException: Failed to publish events at org.apache.flume.sink.kafka.KafkaSink.process(KafkaSink.java:150) at org.apache.flume.sink.DefaultSinkProcessor.process(DefaultSinkProcessor.java:68) at org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:147) at java.lang.Thread.run(Thread.java:745) Caused by: org.apache.flume.ChannelException: Error while getting events from Kafka at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:333) at org.apache.flume.channel.BasicTransactionSemantics.take(BasicTransactionSemantics.java:113) at org.apache.flume.channel.BasicChannelSemantics.take(BasicChannelSemantics.java:95) at org.apache.flume.sink.kafka.KafkaSink.process(KafkaSink.java:97) ... 3 more Caused by: java.lang.InterruptedException at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.reportInterruptAfterWait(AbstractQueuedSynchronizer.java:2014) at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(AbstractQueuedSynchronizer.java:2088) at java.util.concurrent.LinkedBlockingQueue.poll(LinkedBlockingQueue.java:467) at kafka.consumer.ConsumerIterator.makeNext(ConsumerIterator.scala:65) at kafka.consumer.ConsumerIterator.makeNext(ConsumerIterator.scala:33) at

kafka.utils.IteratorTemplate.maybeComputeNext(IteratorTemplate.scala:66) at
kafka.utils.IteratorTemplate.hasNext(IteratorTemplate.scala:58) at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:306) ... 6 more 02 Nov 2016 22:20:06,794
INFO [flume_tbox-topic_SATL2036-1478087994030-54387da2_watcher_executor] (kafka.utils.Logging$class.info:68) - [flume_tbox-
topic_SATL2036-1478087994030-54387da2], stopping watcher executor thread for consumer flume_tbox-topic_SATL2036-
1478087994030-54387da2 02 Nov 2016 22:20:06,816 INFO [flume-channel-tbox-topic_SATL2036-1478087992871-
1a60fb21_watcher_executor] (kafka.utils.Logging$class.info:68) - [flume-channel-tbox-topic_SATL2036-1478087992871-1a60fb21],
stopping watcher executor thread for consumer flume-channel-tbox-topic_SATL2036-1478087992871-1a60fb21 02 Nov 2016
22:20:06,887 WARN [SinkRunner-PollingRunner-FailoverSinkProcessor]
(org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake:332) - Error while getting events from Kafka
java.util.NoSuchElementException at kafka.utils.IteratorTemplate.next(IteratorTemplate.scala:39) at
kafka.consumer.ConsumerIterator.next(ConsumerIterator.scala:46) at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:310) at
org.apache.flume.channel.BasicTransactionSemantics.take(BasicTransactionSemantics.java:113) at
org.apache.flume.channel.BasicChannelSemantics.take(BasicChannelSemantics.java:95) at
org.apache.flume.sink.hdfs.HDFSEventSink.process(HDFSEventSink.java:374) at
org.apache.flume.sink.FailoverSinkProcessor.process(FailoverSinkProcessor.java:182) at
org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:147) at java.lang.Thread.run(Thread.java:745) 02 Nov 2016
22:20:06,888 ERROR [SinkRunner-PollingRunner-FailoverSinkProcessor] (org.apache.flume.sink.hdfs.HDFSEventSink.process:459) -
process failed org.apache.flume.ChannelException: Error while getting events from Kafka at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:333) at
org.apache.flume.channel.BasicTransactionSemantics.take(BasicTransactionSemantics.java:113) at
org.apache.flume.channel.BasicChannelSemantics.take(BasicChannelSemantics.java:95) at
org.apache.flume.sink.hdfs.HDFSEventSink.process(HDFSEventSink.java:374) at
org.apache.flume.sink.FailoverSinkProcessor.process(FailoverSinkProcessor.java:182) at
org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:147) at java.lang.Thread.run(Thread.java:745) Caused by:
java.util.NoSuchElementException at kafka.utils.IteratorTemplate.next(IteratorTemplate.scala:39) at
kafka.consumer.ConsumerIterator.next(ConsumerIterator.scala:46) at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:310) ... 6 more 02 Nov 2016 22:20:06,889
WARN [SinkRunner-PollingRunner-FailoverSinkProcessor] (org.apache.flume.sink.FailoverSinkProcessor.process:185) - Sink k1 failed
and has been sent to failover list org.apache.flume.EventDeliveryException: org.apache.flume.ChannelException: Error while getting
events from Kafka at org.apache.flume.sink.hdfs.HDFSEventSink.process(HDFSEventSink.java:463) at
org.apache.flume.sink.FailoverSinkProcessor.process(FailoverSinkProcessor.java:182) at
org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:147) at java.lang.Thread.run(Thread.java:745) Caused by:
org.apache.flume.ChannelException: Error while getting events from Kafka at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:333) at
org.apache.flume.channel.BasicTransactionSemantics.take(BasicTransactionSemantics.java:113) at
org.apache.flume.channel.BasicChannelSemantics.take(BasicChannelSemantics.java:95) at
org.apache.flume.sink.hdfs.HDFSEventSink.process(HDFSEventSink.java:374) ... 3 more Caused by: java.util.NoSuchElementException
at kafka.utils.IteratorTemplate.next(IteratorTemplate.scala:39) at kafka.consumer.ConsumerIterator.next(ConsumerIterator.scala:46) at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:310) ... 6 more 02 Nov 2016 22:20:06,896
WARN [SinkRunner-PollingRunner-FailoverSinkProcessor]
(org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake:332) - Error while getting events from Kafka
java.util.NoSuchElementException at kafka.utils.IteratorTemplate.next(IteratorTemplate.scala:39) at
kafka.consumer.ConsumerIterator.next(ConsumerIterator.scala:46) at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:310) at
org.apache.flume.channel.BasicTransactionSemantics.take(BasicTransactionSemantics.java:113) at
org.apache.flume.channel.BasicChannelSemantics.take(BasicChannelSemantics.java:95) at
org.apache.flume.sink.hdfs.HDFSEventSink.process(HDFSEventSink.java:374) at
org.apache.flume.sink.FailoverSinkProcessor.process(FailoverSinkProcessor.java:182) at
org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:147) at java.lang.Thread.run(Thread.java:745) 02 Nov 2016
22:20:06,897 ERROR [SinkRunner-PollingRunner-FailoverSinkProcessor] (org.apache.flume.sink.hdfs.HDFSEventSink.process:459) -
process failed org.apache.flume.ChannelException: Error while getting events from Kafka at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:333) at
org.apache.flume.channel.BasicTransactionSemantics.take(BasicTransactionSemantics.java:113) at
org.apache.flume.channel.BasicChannelSemantics.take(BasicChannelSemantics.java:95) at
org.apache.flume.sink.hdfs.HDFSEventSink.process(HDFSEventSink.java:374) at
org.apache.flume.sink.FailoverSinkProcessor.process(FailoverSinkProcessor.java:182) at
org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:147) at java.lang.Thread.run(Thread.java:745) Caused by:
java.util.NoSuchElementException at kafka.utils.IteratorTemplate.next(IteratorTemplate.scala:39) at
kafka.consumer.ConsumerIterator.next(ConsumerIterator.scala:46) at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:310) ... 6 more 02 Nov 2016 22:20:06,898
WARN [SinkRunner-PollingRunner-FailoverSinkProcessor] (org.apache.flume.sink.FailoverSinkProcessor.process:185) - Sink k2 failed
and has been sent to failover list org.apache.flume.EventDeliveryException: org.apache.flume.ChannelException: Error while getting
events from Kafka at org.apache.flume.sink.hdfs.HDFSEventSink.process(HDFSEventSink.java:463) at
org.apache.flume.sink.FailoverSinkProcessor.process(FailoverSinkProcessor.java:182) at
org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:147) at java.lang.Thread.run(Thread.java:745) Caused by:
org.apache.flume.ChannelException: Error while getting events from Kafka at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:333) at
org.apache.flume.channel.BasicTransactionSemantics.take(BasicTransactionSemantics.java:113) at
org.apache.flume.channel.BasicChannelSemantics.take(BasicChannelSemantics.java:95) at
org.apache.flume.sink.hdfs.HDFSEventSink.process(HDFSEventSink.java:374) ... 3 more Caused by: java.util.NoSuchElementException
at kafka.utils.IteratorTemplate.next(IteratorTemplate.scala:39) at kafka.consumer.ConsumerIterator.next(ConsumerIterator.scala:46) at
org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:310) ... 6 more 02 Nov 2016 22:20:06,898
WARN [SinkRunner-PollingRunner-FailoverSinkProcessor]
(org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake:332) - Error while getting events from Kafka

java.util.NoSuchElementException at kafka.utils.IteratorTemplate.next(IteratorTemplate.scala:39) at kafka.consumer.ConsumerIterator.next(ConsumerIterator.scala:46) at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:310) at org.apache.flume.channel.BasicTransactionSemantics.take(BasicTransactionSemantics.java:113) at org.apache.flume.channel.BasicChannelSemantics.take(BasicChannelSemantics.java:95) at org.apache.flume.sink.hdfs.HDFSEventSink.process(HDFSEventSink.java:374) at org.apache.flume.sink.FailoverSinkProcessor.process(FailoverSinkProcessor.java:182) at org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:147) at java.lang.Thread.run(Thread.java:745) 02 Nov 2016 22:20:06,898 ERROR [SinkRunner-PollingRunner-FailoverSinkProcessor] (org.apache.flume.sink.hdfs.HDFSEventSink.process:459) - process failed org.apache.flume.ChannelException: Error while getting events from Kafka at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:333) at org.apache.flume.channel.BasicTransactionSemantics.take(BasicTransactionSemantics.java:113) at org.apache.flume.channel.BasicChannelSemantics.take(BasicChannelSemantics.java:95) at org.apache.flume.sink.hdfs.HDFSEventSink.process(HDFSEventSink.java:374) at org.apache.flume.sink.FailoverSinkProcessor.process(FailoverSinkProcessor.java:182) at org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:147) at java.lang.Thread.run(Thread.java:745) Caused by: java.util.NoSuchElementException at kafka.utils.IteratorTemplate.next(IteratorTemplate.scala:39) at kafka.consumer.ConsumerIterator.next(ConsumerIterator.scala:46) at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:310) ... 6 more 02 Nov 2016 22:20:06,899 WARN [SinkRunner-PollingRunner-FailoverSinkProcessor] (org.apache.flume.sink.FailoverSinkProcessor.process:185) - Sink k3 failed and has been sent to failover list org.apache.flume.EventDeliveryException: org.apache.flume.ChannelException: Error while getting events from Kafka at org.apache.flume.sink.hdfs.HDFSEventSink.process(HDFSEventSink.java:463) at org.apache.flume.sink.FailoverSinkProcessor.process(FailoverSinkProcessor.java:182) at org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:147) at java.lang.Thread.run(Thread.java:745) Caused by: org.apache.flume.ChannelException: Error while getting events from Kafka at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:333) at org.apache.flume.channel.BasicTransactionSemantics.take(BasicTransactionSemantics.java:113) at org.apache.flume.channel.BasicChannelSemantics.take(BasicChannelSemantics.java:95) at org.apache.flume.sink.hdfs.HDFSEventSink.process(HDFSEventSink.java:374) ... 3 more Caused by: java.util.NoSuchElementException at kafka.utils.IteratorTemplate.next(IteratorTemplate.scala:39) at kafka.consumer.ConsumerIterator.next(ConsumerIterator.scala:46) at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doTake(KafkaChannel.java:310) ... 6 more 02 Nov 2016 22:20:06,899 ERROR [SinkRunner-PollingRunner-FailoverSinkProcessor] (org.apache.flume.SinkRunner$PollingRunner.run:160) - Unable to deliver event. Exception follows. org.apache.flume.EventDeliveryException: All sinks failed to process, nothing left to failover to at org.apache.flume.sink.FailoverSinkProcessor.process(FailoverSinkProcessor.java:191) at org.apache.flume.SinkRunner$PollingRunner.run(SinkRunner.java:147) at java.lang.Thread.run(Thread.java:745) 02 Nov 2016 22:20:07,216 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [flume_tbox_parsed_SATL2036-1478088061094-7badc6c0], ZKConsumerConnector shutting down 02 Nov 2016 22:20:07,217 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherManager-1478088061100] Stopping leader finder thread 02 Nov 2016 22:20:07,217 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [flume_tbox_parsed_SATL2036-1478088061094-7badc6c0-leader-finder-thread], Shutting down 02 Nov 2016 22:20:07,218 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [flume_tbox_parsed_SATL2036-1478088061094-7badc6c0-leader-finder-thread], Shutdown completed 02 Nov 2016 22:20:07,218 INFO [flume_tbox_parsed_SATL2036-1478088061094-7badc6c0-leader-finder-thread] (kafka.utils.Logging$class.info:68) - [flume_tbox_parsed_SATL2036-1478088061094-7badc6c0-leader-finder-thread], Stopped 02 Nov 2016 22:20:07,218 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherManager-1478088061100] Stopping all fetchers 02 Nov 2016 22:20:07,218 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume_tbox_parsed_SATL2036-1478088061094-7badc6c0-0-2], Shutting down 02 Nov 2016 22:20:07,219 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume_tbox_parsed_SATL2036-1478088061094-7badc6c0-0-2], Shutdown completed 02 Nov 2016 22:20:07,219 INFO [ConsumerFetcherThread-flume_tbox_parsed_SATL2036-1478088061094-7badc6c0-0-2] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume_tbox_parsed_SATL2036-1478088061094-7badc6c0-0-2], Stopped 02 Nov 2016 22:20:07,220 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume_tbox_parsed_SATL2036-1478088061094-7badc6c0-0-1], Shutting down 02 Nov 2016 22:20:07,220 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume_tbox_parsed_SATL2036-1478088061094-7badc6c0-0-1], Shutdown completed 02 Nov 2016 22:20:07,220 INFO [ConsumerFetcherThread-flume_tbox_parsed_SATL2036-1478088061094-7badc6c0-0-1] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherThread-flume_tbox_parsed_SATL2036-1478088061094-7badc6c0-0-1], Stopped 02 Nov 2016 22:20:07,221 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [ConsumerFetcherManager-1478088061100] All connections stopped 02 Nov 2016 22:20:07,223 INFO [ZkClient-EventThread-48-SATL2036:2181,SATL2037:2181,SATL2038:2181/kafka] (org.I0Itec.zkclient.ZkEventThread.run:82) - Terminate ZkClient event thread. 02 Nov 2016 22:20:07,226 INFO [flume_tbox_parsed_SATL2036-1478088061094-7badc6c0_watcher_executor] (kafka.utils.Logging$class.info:68) - [flume_tbox_parsed_SATL2036-1478088061094-7badc6c0], stopping watcher executor thread for consumer flume_tbox_parsed_SATL2036-1478088061094-7badc6c0 02 Nov 2016 22:20:07,226 INFO [agent-shutdown-hook] (org.apache.zookeeper.ZooKeeper.close:684) - Session: 0x156eb2d4a70421e closed 02 Nov 2016 22:20:07,226 INFO [lifecycleSupervisor-1-0-EventThread] (org.apache.zookeeper.ClientCnxn$EventThread.run:512) - EventThread shut down 02 Nov 2016 22:20:07,227 INFO [agent-shutdown-hook] (kafka.utils.Logging$class.info:68) - [flume_tbox_parsed_SATL2036-1478088061094-7badc6c0], ZKConsumerConnector shut down completed Issue 2: 03 Nov 2016 13:31:29,287 INFO [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.info:68) - Back off for 100 ms before retrying send. Remaining retries = 1 03 Nov 2016 13:31:29,388 INFO [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.info:68) - Fetching metadata from broker id:1,host:SATL2037,port:9092 with correlation id 2307612 for 1 topic(s) Set(channel-tbox-topic) 03 Nov 2016 13:31:29,388 INFO [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.info:68) - Connected to SATL2037:9092 for producing 03 Nov 2016 13:31:29,389 INFO [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.info:68) - Disconnecting from SATL2037:9092 03 Nov 2016 13:31:29,443 INFO [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.info:68) - Connected to SATL2037:9092 for producing 03 Nov 2016 13:31:29,549 INFO [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.info:68) - Disconnecting from SATL2037:9092 03 Nov 2016 13:31:29,550 WARN [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.warn:89) - Failed to send producer request with correlation id 2308613 to broker 2 with data for partitions [channel-tbox-topic,4] java.io.IOException: Connection reset by peer at sun.nio.ch.FileDispatcherImpl.writev0(Native Method) at sun.nio.ch.SocketDispatcher.writev(SocketDispatcher.java:51) at sun.nio.ch.IOUtil.write(IOUtil.java:148) at

sun.nio.ch.SocketChannelImpl.write(SocketChannelImpl.java:504) at java.nio.channels.SocketChannel.write(SocketChannel.java:502) at kafka.network.BoundedByteBufferSend.writeTo(BoundedByteBufferSend.scala:56) at kafka.network.Send$class.writeCompletely(Transmission.scala:75) at kafka.network.BoundedByteBufferSend.writeCompletely(BoundedByteBufferSend.scala:26) at kafka.network.BlockingChannel.send(BlockingChannel.scala:92) at kafka.producer.SyncProducer.liftedTree1$1(SyncProducer.scala:72) at kafka.producer.SyncProducer.kafka$producer$SyncProducer$$doSend(SyncProducer.scala:71) at kafka.producer.SyncProducer$$anonfun$send$1$$anonfun$apply$mcVsp1.applymcVsp(SyncProducer.scala:102) at kafka.producer.SyncProducer$$anonfun$send$1$$anonfun$apply$mcVsp1.apply(SyncProducer.scala:102) at kafka.producer.SyncProducer$$anonfun$send$1$$anonfun$apply$mcVsp1.apply(SyncProducer.scala:102) at kafka.metrics.KafkaTimer.time(KafkaTimer.scala:33) at kafka.producer.SyncProducer$$anonfun$send$1.applymcVsp(SyncProducer.scala:101) at kafka.producer.SyncProducer$$anonfun$send$1.apply(SyncProducer.scala:101) at kafka.producer.SyncProducer$$anonfun$send$1.apply(SyncProducer.scala:101) at kafka.metrics.KafkaTimer.time(KafkaTimer.scala:33) at kafka.producer.SyncProducer.send(SyncProducer.scala:100) at kafka.producer.async.DefaultEventHandler.kafka$producer$async$DefaultEventHandler$$send(DefaultEventHandler.scala:255) at kafka.producer.async.DefaultEventHandler$$anonfun$dispatchSerializedData$2.apply(DefaultEventHandler.scala:106) at kafka.producer.async.DefaultEventHandler$$anonfun$dispatchSerializedData$2.apply(DefaultEventHandler.scala:100) at scala.collection.TraversableLike$WithFilter$$anonfun$foreach$1.apply(TraversableLike.scala:772) at scala.collection.mutable.HashMap$$anonfun$foreach$1.apply(HashMap.scala:98) at scala.collection.mutable.HashMap$$anonfun$foreach$1.apply(HashMap.scala:98) at scala.collection.mutable.HashTable$class.foreachEntry(HashTable.scala:226) at scala.collection.mutable.HashMap.foreachEntry(HashMap.scala:39) at scala.collection.mutable.HashMap.foreach(HashMap.scala:98) at scala.collection.TraversableLike$WithFilter.foreach(TraversableLike.scala:771) at kafka.producer.async.DefaultEventHandler.dispatchSerializedData(DefaultEventHandler.scala:100) at kafka.producer.async.DefaultEventHandler.handle(DefaultEventHandler.scala:72) at kafka.producer.Producer.send(Producer.scala:76) at kafka.javaapi.producer.Producer.send(Producer.scala:42) at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doCommit(KafkaChannel.java:357) at org.apache.flume.channel.BasicTransactionSemantics.commit(BasicTransactionSemantics.java:151) at org.apache.flume.channel.ChannelProcessor.processEventBatch(ChannelProcessor.java:192) at org.apache.flume.source.kafka.KafkaSource.process(KafkaSource.java:130) at org.apache.flume.source.PollableSourceRunner$PollingRunner.run(PollableSourceRunner.java:139) at java.lang.Thread.run(Thread.java:745) 03 Nov 2016 13:31:29,550 INFO [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.info:68) - Back off for 100 ms before retrying send. Remaining retries = 0 03 Nov 2016 13:31:29,651 INFO [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.info:68) - Fetching metadata from broker id:0,host:SATL2036,port:9092 with correlation id 2308614 for 1 topic(s) Set(channel-tbox-topic) 03 Nov 2016 13:31:29,651 INFO [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.info:68) - Connected to SATL2036:9092 for producing 03 Nov 2016 13:31:29,652 INFO [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.info:68) - Disconnecting from SATL2036:9092 03 Nov 2016 13:31:29,653 ERROR [PollableSourceRunner-KafkaSource-r1] (kafka.utils.Logging$class.error:97) - Failed to send requests for topics channel-tbox-topic with correlation ids in [2304607,2308614] 03 Nov 2016 13:31:29,653 WARN [PollableSourceRunner-KafkaSource-r1] (org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doCommit:363) - Sending events to Kafka failed kafka.common.FailedToSendMessageException: Failed to send messages after 3 tries. at kafka.producer.async.DefaultEventHandler.handle(DefaultEventHandler.scala:90) at kafka.producer.Producer.send(Producer.scala:76) at kafka.javaapi.producer.Producer.send(Producer.scala:42) at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doCommit(KafkaChannel.java:357) at org.apache.flume.channel.BasicTransactionSemantics.commit(BasicTransactionSemantics.java:151) at org.apache.flume.channel.ChannelProcessor.processEventBatch(ChannelProcessor.java:192) at org.apache.flume.source.kafka.KafkaSource.process(KafkaSource.java:130) at org.apache.flume.source.PollableSourceRunner$PollingRunner.run(PollableSourceRunner.java:139) at java.lang.Thread.run(Thread.java:745) 03 Nov 2016 13:31:29,653 ERROR [PollableSourceRunner-KafkaSource-r1] (org.apache.flume.source.kafka.KafkaSource.process:153) - KafkaSource EXCEPTION, {} org.apache.flume.ChannelException: Unable to put batch on required channel: org.apache.flume.channel.kafka.KafkaChannel{name: c1} at org.apache.flume.channel.ChannelProcessor.processEventBatch(ChannelProcessor.java:200) at org.apache.flume.source.kafka.KafkaSource.process(KafkaSource.java:130) at org.apache.flume.source.PollableSourceRunner$PollingRunner.run(PollableSourceRunner.java:139) at java.lang.Thread.run(Thread.java:745) Caused by: org.apache.flume.ChannelException: Commit failed as send to Kafka failed at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doCommit(KafkaChannel.java:364) at org.apache.flume.channel.BasicTransactionSemantics.commit(BasicTransactionSemantics.java:151) at org.apache.flume.channel.ChannelProcessor.processEventBatch(ChannelProcessor.java:192) ... 3 more Caused by: kafka.common.FailedToSendMessageException: Failed to send messages after 3 tries. at kafka.producer.async.DefaultEventHandler.handle(DefaultEventHandler.scala:90) at kafka.producer.Producer.send(Producer.scala:76) at kafka.javaapi.producer.Producer.send(Producer.scala:42) at org.apache.flume.channel.kafka.KafkaChannel$KafkaTransaction.doCommit(KafkaChannel.java:357) ... 5 more