Item 338
git_comments:

1. merges can lead to this exception

git_commits:

1. **summary:** SOLR-12412: Fix test failure
   message: SOLR-12412: Fix test failure

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Leader should give up leadership when IndexWriter.tragedy occur
   description: When a leader meets some kind of unrecoverable exception (ie: CorruptedIndexException). The shard will go into the readable state and human has to intervene. In that case, if there are another active replica in the same shard, the leader should give up its leadership.

jira_issues_comments:

1. Does any come up with other critical exceptions beside CorruptedIndexException?
2. Just a thought here, but what about anything inside {{IndexWriter.tragedy}}?
3. Thanks [~tflobbe], that is a good idea. I attached a patch for this ticket. Whenever a request gets failed we will check the {{IndexWriter.tragedy}} if there is another active replica in the same shard and the current replica is the leader (no need for handle the case when the replica is not leader since the leader will bring the replica into recovery state). It will enqueue a DELETE and ADD operation to the Overseer.
4. Final patch with another test and precommit fix. I will commit it soon.
5. Commit 1197717611094c755b271db6e7a8614fe9406bb5e in lucene-solr's branch refs/heads/master from [~caomanhdat] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=1197176] SOLR-12412: Leader should give up leadership when IndexWriter.tragedy occur
6. Commit fddf35cfebd3f612a5e5089e76aa02b105209e6d in lucene-solr's branch refs/heads/branch_7x from [~caomanhdat] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=fddf35c] SOLR-12412: Leader should give up leadership when IndexWriter.tragedy occur
7. Thanks for working on this [~caomanhdat]! I'm wondering if there can be a way to give up leadership that's more light weight than adding/removing replicas while still being safe. Maybe something that ends up doing a core reload?
8. Policeman Jenkins found a reproducing seed [https://jenkins.thetaphi.de/job/Lucene-Solr-7.x-MacOSX/734/] for test failures that {{git bisect}} blames on commit {{fddf35c}} on this issue: {noformat} Checking out Revision 80eb5da7393dd25c8cb566194eb9158de212bfb2 (refs/remotes/origin/branch_7x) [...] [junit4] 2> NOTE: reproduce with: ant test -Dtestcase=TestPullReplica -Dtests.method=testKillLeader -Dtests.seed=89003455250E12D2 -Dtests.slow=true -Dtests.locale=lg -Dtests.timezone=America/Rainy_River -Dtests.asserts=true -Dtests.file.encoding=US-ASCII [junit4] FAILURE 60.4s J1 | TestPullReplica.testKillLeader <<< [junit4] > Throwable #1: java.lang.AssertionError: Replica core_node4 not up to date after 10 seconds expected:<1> but was:<0> [junit4] > at __randomizedtesting.SeedInfo.seed([89003455250E12D2:C016C0E147B58684]:0) [junit4] > at org.apache.solr.cloud.TestPullReplica.waitForNumDocsInAllReplicas(TestPullReplica.java:542) [junit4] > at org.apache.solr.cloud.TestPullReplica.doTestNoLeader(TestPullReplica.java:490) [junit4] > at org.apache.solr.cloud.TestPullReplica.testKillLeader(TestPullReplica.java:309) [junit4] > at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) [junit4] > at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) [junit4] > at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) [junit4] > at java.base/java.lang.reflect.Method.invoke(Method.java:564) [junit4] > at java.base/java.lang.Thread.run(Thread.java:844) [...] [junit4] 2> NOTE: reproduce with: ant test -Dtestcase=TestPullReplica -Dtests.method=testRemoveAllWriterReplicas -Dtests.seed=89003455250E12D2 -Dtests.slow=true -Dtests.locale=lg -Dtests.timezone=America/Rainy_River -Dtests.asserts=true -Dtests.file.encoding=US-ASCII [junit4] FAILURE 24.6s J1 | TestPullReplica.testRemoveAllWriterReplicas <<< [junit4] > Throwable #1: java.lang.AssertionError: Replica core_node4 not up to date after 10 seconds expected:<1> but was:<0> [junit4] > at __randomizedtesting.SeedInfo.seed([89003455250E12D2:1A0EA86E31F0FB7B]:0) [junit4] > at org.apache.solr.cloud.TestPullReplica.waitForNumDocsInAllReplicas(TestPullReplica.java:542) [junit4] > at org.apache.solr.cloud.TestPullReplica.doTestNoLeader(TestPullReplica.java:490) [junit4] > at org.apache.solr.cloud.TestPullReplica.testRemoveAllWriterReplicas(TestPullReplica.java:303) [junit4] > at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) [junit4] > at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) [junit4] > at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) [junit4] > at java.base/java.lang.reflect.Method.invoke(Method.java:564) [junit4] > at java.base/java.lang.Thread.run(Thread.java:844) [...] [junit4] 2> NOTE: test params are: codec=HighCompressionCompressingStoredFields(storedFieldsFormat=CompressingStoredFieldsFormat(compressionMode=HIGH_COMPRESSION, chunkSize=8218, maxDocsPerChunk=6, blockSize=10), termVectorsFormat=CompressingTermVectorsFormat(compressionMode=HIGH_COMPRESSION, chunkSize=8218, blockSize=10)), sim=RandomSimilarity(queryNorm=true): {}, locale=lg, timezone=America/Rainy_River [junit4] 2> NOTE: Mac OS X 10.11.6 x86_64/Oracle Corporation 9 (64-bit)/cpus=3,threads=1,free=262884464,total=536870912 {noformat}
9. Thanks [~steve_rowe], I will take a look at the failure. [~tflobbe] I tried to do that, but it will be quite complex, the process will be (not mention the race condition we can meet) * The core publish itself as DOWN * The core cancel it election context * The core delete its index dir * ... Given that tragic exception is not a frequent event and using Overseer will bring us some benefits like * The update request that met the exception does not get blocked (async) * Much cleaner and well-tested approach * We can easily improve the solution to make it more robust. Ex: when delete replica failed because the node went down, Overseer can remove the replica from clusterstate (therefore even when the node come back, it will be automatically removed) then, Overseer can add a new replica in another node.
10. Commit cd08c7ef13613ceb88c1caf7b25e793ed51d47af in lucene-solr's branch refs/heads/master from [~caomanhdat] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=cd08c7e] SOLR-12412: release IndexWriter after getting tragic exception
11. Commit 0dc6ef996eab378bdd8329153bdecddbf89af9ee in lucene-solr's branch refs/heads/branch_7x from [~caomanhdat] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=0dc6ef9] SOLR-12412: release IndexWriter after getting tragic exception
12. Hi [~caomanhdat] , Maybe we could do something like this here as well to avoid this in the future ? https://issues.apache.org/jira/browse/SOLR-11616?focusedCommentId=16477719&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-16477719 cc [~dsmiley]

13. Yes, +1 to adding a withIndexWriter(lambda) method similar to this guy: org.apache.solr.core.SolrCore#withSearcher This ref-counted business is error-rpone. ~ David
14. Jenkins is reporting quite a few failures for this test. I'm attaching one such run. I ran the seed a couple of times locally but was not able to reproduce it , so it's timing related most likely.
15. Dat: in the past 7 days, LeaderTragicEventTest.testOtherReplicasAreNotActive has failed 36.33% (222 / 611) of all jenkins runs, and LeaderTragicEventTest.test has failed 21.28% (130 / 611). In just the past 24 hours, we've seen a failure rate of 29.09% (16 / 55) for both methods. It seems that even after your most recent commit, these tests need significant hardening to run even remotely close to reliably?
16. This test class has two methods * test() * testOtherReplicasAreNotActive() Both try creating a collection "collection1" . We should probably put the delete collection in a finally block. This would avoid the following error {code:java} [junit4] 2> 13586 INFO (TEST-LeaderTragicEventTest.test-seed#[7146D51E1F1D9F1A]) [ ] o.a.s.SolrTestCaseJ4 ###Starting test [junit4] 2> 13588 INFO (qtp1687913357-34) [n:127.0.0.1:36827_solr] o.a.s.h.a.CollectionsHandler Invoked Collection Action :create with params collection.configName=config&name=collection1&nrtReplicas=2&action=CREATE&numShards=1&wt=javabin&version=2 and sendToOCPQueue=true [junit4] 2> 13590 INFO (OverseerThreadFactory-38-thread-1) [ ] o.a.s.c.a.c.CreateCollectionCmd Create collection collection1 [junit4] 2> 13591 ERROR (OverseerThreadFactory-38-thread-1) [ ] o.a.s.c.a.c.OverseerCollectionMessageHandler Collection: collection1 operation: create failed:org.apache.solr.common.SolrException: collection already exists: collection1 [junit4] 2> at org.apache.solr.cloud.api.collections.CreateCollectionCmd.call(CreateCollectionCmd.java:106) [junit4] 2> at org.apache.solr.cloud.api.collections.OverseerCollectionMessageHandler.processMessage(OverseerCollectionMessageHandler.java:255) [junit4] 2> at org.apache.solr.cloud.OverseerTaskProcessor$Runner.run(OverseerTaskProcessor.java:469) [junit4] 2> at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$execute$0(ExecutorUtil.java:209) [junit4] 2> at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) [junit4] 2> at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) [junit4] 2> at java.lang.Thread.run(Thread.java:748){code} Since testOtherReplicasAreNotActive() failed with an error , it didn't delete the collection1. test() was run after that and hit the above error. test() still passed even if the create collection failed (which means there was already a corrupted index) . Sounds fishy? We could replace this the following line? {code:java} - int numReplicas = random().nextInt(2) + 1; + int numReplicas = TestUtil.nextInt(random(), 1, 2);{code} testOtherReplicasAreNotActive() -> When there are two replicas , where are we actually checking if it becomes active or not after it has been started again? i.e after this statement should we be checking if it becomes active and fail the test? {code:java} if (otherReplicaJetty != null) { // won't be able to do anything here, since this replica can't recovery from the leader otherReplicaJetty.start(); }{code} testOtherReplicasAreNotActive() -> when the test selects one replica , what are we testing exactly ? From what I can understand we are corrupting the leader of a single sharded collection and then validating if it's still the leader ? I'm trying to understand the corruptLeader() method : Why are we trying to delete segment files after every add ? What if we just add the 100 docs and then delete the segments_N file ? Happy to pitch in just wanted to understand the test better before diving in
17. With regards to the actual failure , I think we are shutting down the wrong Jetty? From the seed we have numReplicas=2. Which means we want to shutdown the non-leader shard but from the logs it's shutting down the leader jetty? And then when we go to corrupt the leader jetty , it's actually closed ? {code:java} [junit4] 2> 13526 INFO (TEST-LeaderTragicEventTest.testOtherReplicasAreNotActive-seed#[7146D51E1F1D9F1A]) [ ] o.a.s.c.ZkController Remove node as live in ZooKeeper:/live_nodes/127.0.0.1:35477_solr [junit4] 2> 13526 INFO (TEST-LeaderTragicEventTest.testOtherReplicasAreNotActive-seed#[7146D51E1F1D9F1A]) [ ] o.a.s.m.SolrMetricManager Closing metric reporters for registry=solr.cluster, tag=null [junit4] 2> 13526 INFO (zkCallback-17-thread-1) [ ] o.a.s.c.c.ZkStateReader Updated live nodes from ZooKeeper... (2) -> (1) .... [junit4] 2> 13543 INFO (coreCloseExecutor-33-thread-1) [n:127.0.0.1:35477_solr c:collection1 s:shard1 r:core_node3 x:collection1_shard1_replica_n1] o.a.s.m.SolrMetricManager Closing metric reporters for registry=solr.collection.collection1.shard1.leader, tag=f37433 ... [junit4] 2> 13554 INFO (OverseerStateUpdate-72132540686336006-127.0.0.1:35477_solr-n_0000000000) [ ] o.a.s.c.Overseer Overseer Loop exiting : 127.0.0.1:35477_solr [junit4] 2> 13554 WARN (OverseerAutoScalingTriggerThread-72132540686336006-127.0.0.1:35477_solr-n_0000000000) [ ] o.a.s.c.a.OverseerTriggerThread OverseerTriggerThread woken up but we are closed, exiting. [junit4] 2> 13562 INFO (zkCallback-17-thread-1) [ ] o.a.s.c.OverseerElectionContext I am going to be the leader 127.0.0.1:36827_solr [junit4] 2> 13562 INFO (zkCallback-17-thread-1) [ ] o.a.s.c.Overseer Overseer (id=72132540686336005-127.0.0.1:36827_solr-n_0000000001) starting ... [junit4] 2> 13575 INFO (TEST-LeaderTragicEventTest.testOtherReplicasAreNotActive-seed#[7146D51E1F1D9F1A]) [ ] o.a.s.SolrTestCaseJ4 ###Ending testOtherReplicasAreNotActive [junit4] 2> NOTE: reproduce with: ant test -Dtestcase=LeaderTragicEventTest -Dtests.method=testOtherReplicasAreNotActive -Dtests.seed=7146D51E1F1D9F1A -Dtests.multiplier=3 -Dtests.slow=true -Dtests.badapples=true -Dtests.locale=es-CL -Dtests.timezone=Pacific/Niue -Dtests.asserts=true -Dtests.file.encoding=ISO-8859-1 [junit4] ERROR 5.96s J2 | LeaderTragicEventTest.testOtherReplicasAreNotActive <<< [junit4] > Throwable #1: java.lang.IllegalStateException: Jetty Connector is not open: -2 [junit4] > at __randomizedtesting.SeedInfo.seed([7146D51E1F1D9F1A:F4F2F96923E22682]:0) [junit4] > at org.apache.solr.client.solrj.embedded.JettySolrRunner.getBaseUrl(JettySolrRunner.java:499) [junit4] > at org.apache.solr.cloud.MiniSolrCloudCluster.getReplicaJetty(MiniSolrCloudCluster.java:539) [junit4] > at org.apache.solr.cloud.LeaderTragicEventTest.corruptLeader(LeaderTragicEventTest.java:100) [junit4] > at org.apache.solr.cloud.LeaderTragicEventTest.testOtherReplicasAreNotActive(LeaderTragicEventTest.java:150) [junit4] > at java.lang.Thread.run(Thread.java:748){code}
18. Sorry about the failure, I will take a look today.
19. Commit 705e6f76a44fc774693c36e598022466e0cb1a95 in lucene-solr's branch refs/heads/master from [~caomanhdat] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=705e6f7] SOLR-12412: Fix test failure
20. Commit eed5e7bb1c28fc99982a8d13b33c68425e99e21c in lucene-solr's branch refs/heads/branch_7x from [~caomanhdat] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=eed5e7b] SOLR-12412: Fix test failure
21. Hi Dat, The Jira description reads "When a leader meets some kind of unrecoverable exception (ie: CorruptedIndexException). The shard will go into the readable state and human has to intervene. In that case, it will be the best if the leader gives up its leadership and let other replicas become the leader." But in the test we are asserting this? {code:java} assertEquals(leader.getName(), oldLeader.getName());{code} I had a question that I posted yesterday , reposting it for reference : testOtherReplicasAreNotActive() -> When there are two replicas , where are we actually checking if it becomes active or not after it has been started again? i.e after this statement should we be checking if it becomes active and fail the test? {code:java} if (otherReplicaJetty != null) { // won't be able to do anything here, since this replica can't recovery from the leader otherReplicaJetty.start(); }{code}
22. [~varun] The leader will only give up its leadership only in the case there are another active replica in the same shard. In the {{testOtherReplicasAreNotActive}}, we randomly 2 cases: # A shard with only 1 replica # A shard with 2 replica but the *non leader replica* is DOWN In both case, the leader should not give up its leadership.
23. Got it! {code:java} //TODO better way to test this Thread.sleep(5000); Replica leader = getCollectionState(collection).getSlice("shard1").getLeader(); assertEquals(leader.getName(), oldLeader.getName()); if (otherReplicaJetty != null) { // won't be able to do anything here, since this replica can't recovery from the leader otherReplicaJetty.start(); }{code} Should we start the otherReplicaJetty and then check if the leader doesn't change in the test i.e reverse the order here ? Also maybe we can add the explanation as comments to the test code ? To someone new it would make it a lot easier to understand what this test is trying to do.
24. ASF Jenkins found a reproducing seed for a {{LeaderTragicEventTest}} failure [https://builds.apache.org/job/Lucene-Solr-NightlyTests-7.x/271/]: {noformat} Checking out Revision 950b7b6b1b92849721eaed50ecad9711199180e8 (refs/remotes/origin/branch_7x) [...] [junit4] 2> NOTE: reproduce with: ant test -Dtestcase=LeaderTragicEventTest -Dtests.seed=14F869F052BC897B -Dtests.multiplier=2 -Dtests.nightly=true -Dtests.slow=true -Dtests.linedocsfile=/home/jenkins/jenkins-slave/workspace/Lucene-Solr-NightlyTests-7.x/test-data/enwiki.random.lines.txt -Dtests.locale=de-DE -Dtests.timezone=US/Michigan -Dtests.asserts=true -Dtests.file.encoding=US-ASCII [junit4] ERROR 0.00s J1 | LeaderTragicEventTest (suite) <<< [junit4] > Throwable #1: java.lang.AssertionError: ObjectTracker found 1 object(s) that were not released!!! [TransactionLog] [junit4] > org.apache.solr.common.util.ObjectReleaseTracker$ObjectTrackerException: org.apache.solr.update.TransactionLog [junit4] > at org.apache.solr.common.util.ObjectReleaseTracker.track(ObjectReleaseTracker.java:42) [junit4] > at org.apache.solr.update.TransactionLog.<init>(TransactionLog.java:188) [junit4] > at

org.apache.solr.update.UpdateLog.newTransactionLog(UpdateLog.java:467) [junit4] > at
org.apache.solr.update.UpdateLog.ensureLog(UpdateLog.java:1323) [junit4] > at org.apache.solr.update.UpdateLog.add(UpdateLog.java:571) [junit4]
> at org.apache.solr.update.UpdateLog.add(UpdateLog.java:551) [junit4] > at
org.apache.solr.update.DirectUpdateHandler2.doNormalUpdate(DirectUpdateHandler2.java:345) [junit4] > at
org.apache.solr.update.DirectUpdateHandler2.addDoc0(DirectUpdateHandler2.java:283) [junit4] > at
org.apache.solr.update.DirectUpdateHandler2.addDoc(DirectUpdateHandler2.java:233) [junit4] > at
org.apache.solr.update.processor.RunUpdateProcessor.processAdd(RunUpdateProcessorFactory.java:67) [junit4] > at
org.apache.solr.update.processor.UpdateRequestProcessor.processAdd(UpdateRequestProcessor.java:55) [junit4] > at
org.apache.solr.update.processor.DistributedUpdateProcessor.doLocalAdd(DistributedUpdateProcessor.java:951) [junit4] > at
org.apache.solr.update.processor.DistributedUpdateProcessor.versionAdd(DistributedUpdateProcessor.java:1167) [junit4] > at
org.apache.solr.update.processor.DistributedUpdateProcessor.processAdd(DistributedUpdateProcessor.java:634) [junit4] > at
org.apache.solr.update.processor.LogUpdateProcessorFactory$LogUpdateProcessor.processAdd(LogUpdateProcessorFactory.java:103) [junit4] > at
org.apache.solr.handler.loader.JavabinLoader$1.update(JavabinLoader.java:98) [junit4] > at
org.apache.solr.client.solrj.request.JavaBinUpdateRequestCodec$1.readOuterMostDocIterator(JavaBinUpdateRequestCodec.java:188) [junit4] > at
org.apache.solr.client.solrj.request.JavaBinUpdateRequestCodec$1.readIterator(JavaBinUpdateRequestCodec.java:144) [junit4] > at
org.apache.solr.common.util.JavaBinCodec.readObject(JavaBinCodec.java:311) [junit4] > at
org.apache.solr.common.util.JavaBinCodec.readVal(JavaBinCodec.java:256) [junit4] > at
org.apache.solr.client.solrj.request.JavaBinUpdateRequestCodec$1.readNamedList(JavaBinUpdateRequestCodec.java:130) [junit4] > at
org.apache.solr.common.util.JavaBinCodec.readObject(JavaBinCodec.java:276) [junit4] > at
org.apache.solr.common.util.JavaBinCodec.readVal(JavaBinCodec.java:256) [junit4] > at
org.apache.solr.common.util.JavaBinCodec.unmarshal(JavaBinCodec.java:178) [junit4] > at
org.apache.solr.client.solrj.request.JavaBinUpdateRequestCodec.unmarshal(JavaBinUpdateRequestCodec.java:195) [junit4] > at
org.apache.solr.handler.loader.JavabinLoader.parseAndLoadDocs(JavabinLoader.java:109) [junit4] > at
org.apache.solr.handler.loader.JavabinLoader.load(JavabinLoader.java:55) [junit4] > at
org.apache.solr.handler.UpdateRequestHandler$1.load(UpdateRequestHandler.java:97) [junit4] > at
org.apache.solr.handler.ContentStreamHandlerBase.handleRequestBody(ContentStreamHandlerBase.java:68) [junit4] > at
org.apache.solr.handler.RequestHandlerBase.handleRequest(RequestHandlerBase.java:199) [junit4] > at
org.apache.solr.core.SolrCore.execute(SolrCore.java:2541) [junit4] > at org.apache.solr.servlet.HttpSolrCall.execute(HttpSolrCall.java:709) [junit4] >
at org.apache.solr.servlet.HttpSolrCall.call(HttpSolrCall.java:515) [junit4] > at
org.apache.solr.servlet.SolrDispatchFilter.doFilter(SolrDispatchFilter.java:377) [junit4] > at
org.apache.solr.servlet.SolrDispatchFilter.doFilter(SolrDispatchFilter.java:323) [junit4] > at
org.eclipse.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1642) [junit4] > at
org.apache.solr.client.solrj.embedded.JettySolrRunner$DebugFilter.doFilter(JettySolrRunner.java:139) [junit4] > at
org.eclipse.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1642) [junit4] > at
org.eclipse.jetty.servlet.ServletHandler.doHandle(ServletHandler.java:533) [junit4] > at
org.eclipse.jetty.server.handler.ScopedHandler.nextHandle(ScopedHandler.java:255) [junit4] > at
org.eclipse.jetty.server.session.SessionHandler.doHandle(SessionHandler.java:1595) [junit4] > at
org.eclipse.jetty.server.handler.ScopedHandler.nextHandle(ScopedHandler.java:255) [junit4] > at
org.eclipse.jetty.server.handler.ContextHandler.doHandle(ContextHandler.java:1317) [junit4] > at
org.eclipse.jetty.server.handler.ScopedHandler.nextScope(ScopedHandler.java:203) [junit4] > at
org.eclipse.jetty.servlet.ServletHandler.doScope(ServletHandler.java:473) [junit4] > at
org.eclipse.jetty.server.session.SessionHandler.doScope(SessionHandler.java:1564) [junit4] > at
org.eclipse.jetty.server.handler.ScopedHandler.nextScope(ScopedHandler.java:201) [junit4] > at
org.eclipse.jetty.server.handler.ContextHandler.doScope(ContextHandler.java:1219) [junit4] > at
org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:144) [junit4] > at
org.eclipse.jetty.server.handler.gzip.GzipHandler.handle(GzipHandler.java:674) [junit4] > at
org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java:132) [junit4] > at org.eclipse.jetty.server.Server.handle(Server.java:531)
[junit4] > at org.eclipse.jetty.server.HttpChannel.handle(HttpChannel.java:352) [junit4] > at
org.eclipse.jetty.server.HttpConnection.onFillable(HttpConnection.java:260) [junit4] > at
org.eclipse.jetty.io.AbstractConnection$ReadCallback.succeeded(AbstractConnection.java:281) [junit4] > at
org.eclipse.jetty.io.FillInterest.fillable(FillInterest.java:102) [junit4] > at org.eclipse.jetty.io.ssl.SslConnection.onFillable(SslConnection.java:291)
[junit4] > at org.eclipse.jetty.io.ssl.SslConnection$3.succeeded(SslConnection.java:151) [junit4] > at
org.eclipse.jetty.io.FillInterest.fillable(FillInterest.java:102) [junit4] > at org.eclipse.jetty.io.ChannelEndPoint$2.run(ChannelEndPoint.java:118)
[junit4] > at org.eclipse.jetty.util.thread.strategy.EatWhatYouKill.runTask(EatWhatYouKill.java:333) [junit4] > at
org.eclipse.jetty.util.thread.strategy.EatWhatYouKill.doProduce(EatWhatYouKill.java:310) [junit4] > at
org.eclipse.jetty.util.thread.strategy.EatWhatYouKill.tryProduce(EatWhatYouKill.java:168) [junit4] > at
org.eclipse.jetty.util.thread.strategy.EatWhatYouKill.run(EatWhatYouKill.java:126) [junit4] > at
org.eclipse.jetty.util.thread.ReservedThreadExecutor$ReservedThread.run(ReservedThreadExecutor.java:366) [junit4] > at
org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.java:762) [junit4] > at
org.eclipse.jetty.util.thread.QueuedThreadPool$2.run(QueuedThreadPool.java:680) [junit4] > at java.lang.Thread.run(Thread.java:748) [junit4] > at
__randomizedtesting.SeedInfo.seed([14F869F052BC897B]:0) [junit4] > at
org.apache.solr.SolrTestCaseJ4.teardownTestCases(SolrTestCaseJ4.java:304) [junit4] > at java.lang.Thread.run(Thread.java:748) [junit4] Completed
[125/825 (1!)] on J1 in 55.18s, 2 tests, 1 failure <<< FAILURES! {noformat}

25. Hi Dat, Checking again with the doubts that I had regarding this test First question {quote}I'm trying to understand the corruptLeader() method : Why are we trying to delete segment files after every add ? What if we just add the 100 docs and then delete the segments_N file ? {quote} Second question {quote}Should we start the otherReplicaJetty and then check if the leader doesn't change in the test i.e reverse the order here ? {quote} The reason I ask this again is to me what's the point of starting the jetty and putting a code comment that the shard won't be able to do anything without actually validating it? {code:java} if (otherReplicaJetty != null) { // won't be able to do anything here, since this replica can't recovery from the leader otherReplicaJetty.start(); }{code}
26. Hi [~varun] , 1st question: this is the only way to avoid the cache of the directory and trigger merge reliable -> tragic event will reliably occur. You can try to change the code to your strategy and valid that tragic event, in that case, won't occur reliably. 2nd: Yeah, I plan to do that, but too busy with other stuff and it only makes the test less clear. not affect the case I want to test. That comment mean, we won't be able to recover shard to come active, since the leader is already corrupted hence the replica won't be able to do recovery.
27. Commit eada799f576a2a1cb6dd16179a34ef283cdb4101 in lucene-solr's branch refs/heads/master from [~caomanhdat] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=eada799] SOLR-12412: Leak transaction log on tragic event
28. Commit 0dc124aa78e2a1c121a9634e69f84c8b1f6be331 in lucene-solr's branch refs/heads/master from [~caomanhdat] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=0dc124a] SOLR-12412: Fix precommit
29. Commit 41028dc989bc53717878123c0ea3effbbd7351ae in lucene-solr's branch refs/heads/branch_7x from [~caomanhdat] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=41028dc] SOLR-12412: Leak transaction log on tragic event
30. Commit a9f129190f9065c8775a628df181fb53248db488 in lucene-solr's branch refs/heads/branch_7x from [~caomanhdat] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=a9f1291] SOLR-12412: Fix precommit