Item 62
git_comments:

1. through out the test, every shard shuld have 2 replicas, one on each of these two nodes
2. sanity check test setup
3. short circut if our slice isn't active, or has wrong # replicas
4. short circut if collection is deleted or we some how have the wrong number of (active) slices
5. short circut if collection is deleted or we some how have the wrong number of slices
6. now the main check we care about: were the replicas split up on the expected nodes...
7. Note: we're checking all slices, even the inactive (split) slice...
8. **comment:** Note: only 1 slices, but simpler to loop then extract...
   label: code-design
9. make sure our replicas are fully live...
10. **comment:** short circut if our slice has wrong # replicas
    label: code-design

git_commits:

1. **summary:** Harden TestPolicyCloud
   message: Harden TestPolicyCloud - ensure all collections/replicas are active - tighten assertions around expected replica locations - eliminate some redundent code These changes should also help ensure we don't get (more) spurious failures due to SOLR-13616
   label: code-design

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Possible racecondition/deadlock between collection DELETE and PrepRecovery ? (TestPolicyCloud failures)
   description: Based on some recent jenkins failures in TestPolicyCloud, I suspect there is a possible deadlock condition when attempting to delete a collection while recovery is in progress. I haven't been able to identify exactly where/why/how the problem occurs, but it does not appear to be a test specific problem, and seems like it could potentially affect anyone unlucky enough to issue poorly timed DELETE. Details to follow in comments...

jira_issues_comments:

1. **body:** Unlike most tests that explicitly waitFor/assert *active* replicas, TestPolicyCloud (currently) has several tests that only assert the quanity and location of a replica – it doesn't wait for them to become active, so when testing an ADDREPLICA or a SPLITSHARD, those new replicas are still in recovery (or PrepRecovery) when the test tries to do cleanup and delete the collection – which frequently fails with timeout problems. While we can certainly "improve" TestPolicyCloud to wait for recoveries to finish, and all replicas to be active before attempting to delete the collection, a better question is why this is needed? I'm attaching {{thetaphi_Lucene-Solr-master-Linux_24358.log.txt}} which demonstrates the problem in {{TestPolicyCloud.testCreateCollectionAddReplica}} here are some highlights... {noformat} # thetaphi_Lucene-Solr-master-Linux_24358.log.txt # # testCreateCollectionAddReplica # bulk of test logic is finished, test has added a replica and confirmed it's on the expected node # # but meanwhile, recovery is still ongoing... [junit4] 2> 959699 INFO (recoveryExecutor-5888-thread-1-processing-n:127.0.0.1:42097_solr x:testCreateCollectionAddReplica_shard1_replica_n3 c:testCreateCollectionAddReplica s:shard1 r:core_node4) [n:127.0.0.1:42097_solr c:testCreateCollectionAddReplica s:shard1 r:core_node4 x:testCreateCollectionAddReplica_shard1_replica_n3] o.a.s.c.RecoveryStrategy Sending prep recovery command to [https://127.0.0.1:42097/solr]; [WaitForState: action=PREPRECOVERY&core=testCreateCollectionAddReplica_shard1_replica_n1&nodeName=127.0.0.1:42097_solr&coreNodeName=core_node4&state=reco [junit4] 2> 959701 INFO (qtp531873617-17025) [n:127.0.0.1:42097_solr x:testCreateCollectionAddReplica_shard1_replica_n1] o.a.s.h.a.PrepRecoveryOp Going to wait for coreNodeName: core_node4, state: recovering, checkLive: true, onlyIfLeader: true, onlyIfLeaderActive: true [junit4] 2> 959701 INFO (qtp531873617-17025) [n:127.0.0.1:42097_solr x:testCreateCollectionAddReplica_shard1_replica_n1] o.a.s.h.a.PrepRecoveryOp In WaitForState(recovering): collection=testCreateCollectionAddReplica, shard=shard1, thisCore=testCreateCollectionAddReplica_shard1_replica_n1, leaderDoesNotNeedRecovery=false, isLeader? true, live=true, checkLive=true, currentState=down, localState=active, nodeName=127.0.0.1:42097_solr, coreNodeName=core_node4, onlyIfActiveCheckResult=false, nodeProps: core_node4:{ [junit4] 2> "core":"testCreateCollectionAddReplica_shard1_replica_n3", [junit4] 2> "base_url":"https://127.0.0.1:42097/solr", [junit4] 2> "state":"down", [junit4] 2> "node_name":"127.0.0.1:42097_solr", [junit4] 2> "type":"NRT"} ... # the test thread moves on to @After method which calls MiniSolrCloudCluster.deleteAllCollections() ... ... [junit4] 2> 959703 INFO (qtp531873617-17021) [n:127.0.0.1:42097_solr] o.a.s.h.a.CollectionsHandler Invoked Collection Action :delete with params name=testCreateCollectionAddReplica&action=DELETE&wt=javabin&version=2 and sendToOCPQueue=true ... [junit4] 2> 959709 INFO (OverseerThreadFactory-5345-thread-5-processing-n:127.0.0.1:44991_solr) [n:127.0.0.1:44991_solr] o.a.s.c.a.c.OverseerCollectionMessageHandler Executing Collection Cmd=action=UNLOAD&deleteInstanceDir=true&deleteDataDir=true&deleteMetricsHistory=true, asyncId=null ... [junit4] 2> 959750 INFO (qtp531873617-17325) [n:127.0.0.1:42097_solr x:testCreateCollectionAddReplica_shard1_replica_n1] o.a.s.c.SolrCore [testCreateCollectionAddReplica_shard1_replica_n1] CLOSING SolrCore org.apache.solr.core.SolrCore@39444e66 ... [junit4] 2> 959753 INFO (qtp531873617-17325) [n:127.0.0.1:42097_solr x:testCreateCollectionAddReplica_shard1_replica_n1] o.a.s.s.HttpSolrCall [admin] webapp=null path=/admin/cores params={deleteInstanceDir=true&deleteMetricsHistory=true&core=testCreateCollectionAddReplica_shard1_replica_n1&qt=/admin/cores&deleteDataDir=true&action=UNI status=0 QTime=17 # but meanwhile, PrepRecoveryOp is currently blocked on a call to ZkStateReader.waitForState # looking for specific conditions for the leader and (new) replica (that needs to recover) # ... BUT!... the leader core has already been closed, so the watcher never succeeds, # ...so PrepRecovery keeps waitForState ... [junit4] 2> 959855 WARN (watches-5915-thread-1) [ ] o.a.s.c.c.ZkStateReader Error on calling watcher [junit4] 2> => org.apache.solr.common.SolrException: core not found:testCreateCollectionAddReplica_shard1_replica_n1 [junit4] 2> at org.apache.solr.handler.admin.PrepRecoveryOp.lambda$execute$0(PrepRecoveryOp.java:83) [junit4] 2> org.apache.solr.common.SolrException: core not found:testCreateCollectionAddReplica_shard1_replica_n1 [junit4] 2> at org.apache.solr.handler.admin.PrepRecoveryOp.lambda$execute$0(PrepRecoveryOp.java:83) ~[java/:?] [junit4] 2> at org.apache.solr.common.cloud.ZkStateReader.lambda$waitForState$13(ZkStateReader.java:1742) ~[java/:?] [junit4] 2> at org.apache.solr.common.cloud.ZkStateReader$DocCollectionAndLiveNodesWatcherWrapper.onStateChanged(ZkStateReader.java:2309) ~[java/:?] [junit4] 2> at org.apache.solr.common.cloud.ZkStateReader$Notification.run(ZkStateReader.java:2041) ~[java/:?] [junit4] 2> at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:515) ~[?:?] [junit4] 2> at java.util.concurrent.FutureTask.run(FutureTask.java:264) ~[?:?] [junit4] 2> at org.apache.solr.common.util.ExecutorUtil$MDCAwareThreadPoolExecutor.lambda$execute$0(ExecutorUtil.java:209) ~[java/:?] [junit4] 2> at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1128) ~[?:?] [junit4] 2> at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628) ~[?:?] [junit4] 2> at java.lang.Thread.run(Thread.java:835) [?:?] # ... repeats 3 times, same millisecond, diff threads # (watches-5915-thread-2 & watches-5915-thread-3 # Other then some intranode /admin/metrics requests, *NOTHING* else happens for ~90 seconds # Until finally the DELETE collection request times out... [junit4] 2> 1049792 INFO (TEST-TestPolicyCloud.testCreateCollectionAddReplica-seed#[CA8FB8D61B016EFE]) [ ] o.a.s.SolrTestCaseJ4 ###Ending testCreateCollectionAddReplica [junit4] 2> NOTE: reproduce with: ant test -Dtestcase=TestPolicyCloud -Dtests.method=testCreateCollectionAddReplica -Dtests.seed=CA8FB8D61B016EFE -Dtests.multiplier=3 -Dtests.slow=true -Dtests.locale=yi-001 -Dtests.timezone=America/Indiana/Indianapolis -Dtests.asserts=true -Dtests.file.encoding=ISO-8859-1 [junit4] ERROR 92.6s J0 | TestPolicyCloud.testCreateCollectionAddReplica <<< [junit4] > Throwable #1: org.apache.solr.client.solrj.SolrServerException:

Timeout occurred while waiting response from server at: https://127.0.0.1:42097/solr [junit4] > at __randomizedtesting.SeedInfo.seed([CA8FB8D61B016EFE:4AAFDDF80A428658]:0) [junit4] > at org.apache.solr.client.solrj.impl.HttpSolrClient.executeMethod(HttpSolrClient.java:667) [junit4] > at org.apache.solr.client.solrj.impl.HttpSolrClient.request(HttpSolrClient.java:262) [junit4] > at org.apache.solr.client.solrj.impl.HttpSolrClient.request(HttpSolrClient.java:245) [junit4] > at org.apache.solr.client.solrj.impl.LBSolrClient.doRequest(LBSolrClient.java:368) [junit4] > at org.apache.solr.client.solrj.impl.LBSolrClient.request(LBSolrClient.java:296) [junit4] > at org.apache.solr.client.solrj.impl.BaseCloudSolrClient.sendRequest(BaseCloudSolrClient.java:1128) [junit4] > at org.apache.solr.client.solrj.impl.BaseCloudSolrClient.requestWithRetryOnStaleState(BaseCloudSolrClient.java:897) [junit4] > at org.apache.solr.client.solrj.impl.BaseCloudSolrClient.request(BaseCloudSolrClient.java:829) [junit4] > at org.apache.solr.client.solrj.SolrRequest.process(SolrRequest.java:211) [junit4] > at org.apache.solr.client.solrj.SolrRequest.process(SolrRequest.java:228) [junit4] > at org.apache.solr.cloud.MiniSolrCloudCluster.deleteAllCollections(MiniSolrCloudCluster.java:547) [junit4] > at org.apache.solr.cloud.autoscaling.TestPolicyCloud.after(TestPolicyCloud.java:87) [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:567) [junit4] > at java.base/java.lang.Thread.run(Thread.java:835) [junit4] > Caused by: java.net.SocketTimeoutException: Read timed out [junit4] > at java.base/java.net.SocketInputStream.socketRead0(Native Method) [junit4] > at java.base/java.net.SocketInputStream.socketRead(SocketInputStream.java:115) [junit4] > at java.base/java.net.SocketInputStream.read(SocketInputStream.java:168) [junit4] > at java.base/java.net.SocketInputStream.read(SocketInputStream.java:140) [junit4] > at java.base/sun.security.ssl.SSLSocketInputRecord.read(SSLSocketInputRecord.java:448) [junit4] > at java.base/sun.security.ssl.SSLSocketInputRecord.bytesInCompletePacket(SSLSocketInputRecord.java:68) [junit4] > at java.base/sun.security.ssl.SSLSocketImpl.readApplicationRecord(SSLSocketImpl.java:1132) [junit4] > at java.base/sun.security.ssl.SSLSocketImpl$AppInputStream.read(SSLSocketImpl.java:828) {noformat} This pattern of "RECOVERY happens concurrently with DELETE, lots of wasted time with nothing logged, DELETE times out or fails" repeats in other jenkins jobs, and in other test methods like {{testCreateCollectionSplitShard}} ... when these failures happen, the effects sometimes bleed over into toehr test methods – aparently because the old collection is "partially" deleted – the cores are unloaded, but the collection still exists in the cluster state causing problems when future invokations of the {{@After}} method calls {{MiniSolrCloudCluster.deleteAllCollections()}} ... {noformat} [junit4] 2> 1081521 INFO (TEST-TestPolicyCloud.testDataProvider-seed#[CA8FB8D61B016EFE]) [ ] o.a.s.SolrTestCaseJ4 ###Ending testDataProvider [junit4] 2> NOTE: reproduce with: ant test -Dtestcase=TestPolicyCloud -Dtests.method=testDataProvider -Dtests.seed=CA8FB8D61B016EFE -Dtests.multiplier=3 -Dtests.slow=true -Dtests.locale=yi-001 -Dtests.timezone=America/Indiana/Indianapolis -Dtests.asserts=true -Dtests.file.encoding=ISO-8859-1 [junit4] ERROR 31.7s J0 | TestPolicyCloud.testDataProvider <<< [junit4] > Throwable #1: org.apache.solr.client.solrj.impl.HttpSolrClient$RemoteSolrException: Error from server at https://127.0.0.1:42097/solr: Could not find collection : testCreateCollectionAddReplica [junit4] > at __randomizedtesting.SeedInfo.seed([CA8FB8D61B016EFE:F22410E3A9C36A27]:0) [junit4] > at org.apache.solr.client.solrj.impl.HttpSolrClient.executeMethod(HttpSolrClient.java:656) [junit4] > at org.apache.solr.client.solrj.impl.HttpSolrClient.request(HttpSolrClient.java:262) [junit4] > at org.apache.solr.client.solrj.impl.HttpSolrClient.request(HttpSolrClient.java:245) [junit4] > at org.apache.solr.client.solrj.impl.LBSolrClient.doRequest(LBSolrClient.java:368) [junit4] > at org.apache.solr.client.solrj.impl.LBSolrClient.request(LBSolrClient.java:296) [junit4] > at org.apache.solr.client.solrj.impl.BaseCloudSolrClient.sendRequest(BaseCloudSolrClient.java:1128) [junit4] > at org.apache.solr.client.solrj.impl.BaseCloudSolrClient.requestWithRetryOnStaleState(BaseCloudSolrClient.java:897) [junit4] > at org.apache.solr.client.solrj.impl.BaseCloudSolrClient.request(BaseCloudSolrClient.java:829) [junit4] > at org.apache.solr.client.solrj.SolrRequest.process(SolrRequest.java:211) [junit4] > at org.apache.solr.client.solrj.SolrRequest.process(SolrRequest.java:228) [junit4] > at org.apache.solr.cloud.MiniSolrCloudCluster.deleteAllCollections(MiniSolrCloudCluster.java:547) [junit4] > at org.apache.solr.cloud.autoscaling.TestPolicyCloud.after(TestPolicyCloud.java:87) {noformat} FWIW, I should note: * I've been unable to reproduce this locally * the failures happen on diff jenkins servers, on diff branches, and different OSes * so far all of the failures i've seen occur when randomized ssl=true, but AFAICT there isn't any obvious indicatiion that the problem relates to the use of ssl – i think that just that the added CPU contention caused by using SSL seems to slow things down enough on the jenkins machine that the race contidion happens ** ie: because of SSL slowdown, the recovery isn't fully completed before the DELETE collection cleanup requests are processed ---- My initial hunch/impression was that Overseer (Queue) was getting deadlocked trying to invoke the PrepRecoveryOp (which was blocked waiting for a leader that no longer exists) preventing the DeleteCollectionCmd from executing (which would remove the collection from the ClusterState – the only state change possible to cause the PrepRecoveryOp's waitForState call to abort. I tried to write a new test forcing this specific situation (attached), using a {{CloseHook}} on the leader {{SolrCore}} and {{waitForState()}} predicate that would gate eachother using {{CountDownLatches}} to try and coerce the specific sequence of events that seemed to be problematic – but i can't seem to force a this type of failure (or even semi-reliably trigger it given that the ZkStateReader executes watchers in a random order). If anyone else has any ideas what's going on here i'm all ears ... in the meantime I'm going to try and work up some general improvements to TestPolicyCLoud to make the assertions tighter and insure all the expected replicas are fully active before finishing the test.
label: code-design

2. Hi [~hossman], thanks a lot for taking a look at this failure. Your analysis make tracking down things much faster. I think the problem here belongs to {{ZkStateReader}} (or the way we use it in {{PreRecoveryOp}}). In {{PreRecoveryOp}} (this call was added/refactored in SOLR-12801) {code} coreContainer.getZkController().getZkStateReader().waitForState(collectionName, conflictWaitMs, TimeUnit.MILLISECONDS, (n, c) -> { try (SolrCore core = coreContainer.getCore(cname)) { if (core == null) throw new SolrException(SolrException.ErrorCode.BAD_REQUEST, "core not found:" + cname); ... {code} So the idea here is we throw an SolrException whenever the core (leader) is not found, so the method will terminate and return response to the replica which is doing recovery. But ZkStateReader will never throw any exception happened in Watcher (CollectionStatePredicate) to the upper level, it just log out the exception and continue. https://github.com/apache/lucene-solr/blob/fb30ded6436a577af86e5db201eed01170102a97/solr/solrj/src/java/org/apache/solr/common/cloud/ZkStateReader.java#L2045 Furthermore that exception watcher never get removed from the list of watchers, so whenever clusterstate get changed, this method will get invoked -> that is the reason why that {{"Error on calling watcher"}} appeared multiple times.
3. **body:** I'm not sure we should change the {{waitForState}} logic to rethrow Exceptions or revert back PrepRecoveryOp to its previous version (before SOLR-12801). Need more thought about it.
label: code-design
4. Hoss and Dat -- thank you for investigating this! All usages of CollectionStateWatcher or LiveNodesWatcher will suffer from this problem i.e. the thread that runs the watcher swallows the exception so we should audit all their usages regardless of what solution we go for.
5. **body:** {quote}I'm not sure we should change the waitForState logic to rethrow Exceptions or revert back PrepRecoveryOp to its previous version ... {quote} {quote}Hoss and Dat – thank you for investigating this! All usages of CollectionStateWatcher or LiveNodesWatcher will suffer from this problem i.e. the thread that runs the watcher swallows the exception ... {quote} Well, generally speaking there isn't any way (i can think of) for the thread executing a Watcher to do anything _but_ swallow any exceptions from the watcher – it can't propogated it back to the "caller" of registrWatcher or anything like that .. if the caller wanted to be informed then the Watcher it registered should be catching the exceptions itself. But to Dat's point: in the specific case of {{waitForState}} – there ZkStateReader *is* creating it's own Watcher to wrap the input Predicate, and we could in fact make waitForState do something inside that Watcher that catches any Exception thrown by the Predicate and short circuts out of the {{waitForState}} call, wrapping/re-throwing the exception in the meantime. But those seem like "broader" problems with regards to where/how the different callers are using the Watcher/waitForState APIs that we should probably create a new issue to track (for auditing all of them and clarifying the behavior in the javadocs) ... frankly i think in this specific jira we should be asking a lot more questions about the _specific_ predicate used in PrepForRecovery's waitForState call ... notably what exactly is the expectation here when the SolrCore (that prepRecovery wants to recover from) can't be found _in the local CoreContainer_ ... deleting the collection is just one example, are there other situations where the core may not be found at this point in the code? (node shutdown perhaps? autoscaling removing a replica) ? what about a few lines later... {code:java} if (onlyIfLeader != null && onlyIfLeader) { if (!core.getCoreDescriptor().getCloudDescriptor().isLeader()) { throw new SolrException(SolrException.ErrorCode.BAD_REQUEST, "We are not the leader"); } } {code} ...even if the SolrCore is found, if we expect it to be the shard leader, and it's not (what if there has beena leader election in the meantime?) then that's another type of problem that will also cause the predicate to throw an exception that will (aparently) cause PrepRecovery to stall. what should PrepRecovery do here? i suspect that in general the use of waitForState here in PrepRecoveryOp is "ok in concept" ... we just need to make the predicate

smarter about exiting immeidately in these situations instead of throwing an exception that gets swallowed ... i'm just not sure what the right behavior for PrepRecovery *is* in these sitautions. ---- I don't suppose either of you were able to spot what's "wrong" with my test that it doesn't force a failure in this situation?
label: code-design

6. Commit 8a277cab7d15c03ff59577efceb6a0cb281d095a in lucene-solr's branch refs/heads/master from Chris M. Hostetter [https://gitbox.apache.org/repos/asf?p=lucene-solr.git;h=8a277ca] Harden TestPolicyCloud - ensure all collections/replicas are active - tighten assertions around expected replica locations - eliminate some redundent code These changes should also help ensure we don't get (more) spurious failures due to SOLR-13616
7. not sure why/how gitbox missed the 8x cherry-pick: https://gitbox.apache.org/repos/asf?p=lucene-solr.git;h=81b2e06ffe6bddcd8d25b24c79683281da85baee
8. Commit 8a277cab7d15c03ff59577efceb6a0cb281d095a in lucene-solr's branch refs/heads/jira/SOLR-13565 from Chris M. Hostetter [https://gitbox.apache.org/repos/asf?p=lucene-solr.git;h=8a277ca] Harden TestPolicyCloud - ensure all collections/replicas are active - tighten assertions around expected replica locations - eliminate some redundent code These changes should also help ensure we don't get (more) spurious failures due to SOLR-13616
9. Commit 7ddba3b7123f11d387ff395516cf46c607ac21ee in lucene-solr's branch refs/heads/master from Chris M. Hostetter [https://gitbox.apache.org/repos/asf?p=lucene-solr.git;h=7ddba3b] Harden DeleteReplicaTest * tighten assertions related to type of watcher that should be removed * use waitForActiveCollection before deleting collections to work around SOLR-13616 and/or SOLR-13627
10. Commit d91900a4a271829325a948aff874301979aba202 in lucene-solr's branch refs/heads/branch_8x from Chris M. Hostetter [https://gitbox.apache.org/repos/asf?p=lucene-solr.git;h=d91900a] Harden DeleteReplicaTest * tighten assertions related to type of watcher that should be removed * use waitForActiveCollection before deleting collections to work around SOLR-13616 and/or SOLR-13627 (cherry picked from commit 7ddba3b7123f11d387ff395516cf46c607ac21ee)
11. Commit 32da33936532a13523be522a8e86820c5bd9a497 in lucene-solr's branch refs/heads/branch_8x from Chris M. Hostetter [https://gitbox.apache.org/repos/asf?p=lucene-solr.git;h=32da339] Harden RulesTest * ensure all collections/replicas are active * use waitForState or waitForActiveCollection before checking rules/snitch to prevent false failures on stale state * ensure cluster policy is cleared after each test method Some of these changes should also help ensure we don't get (more) spurious failures due to SOLR-13616 (cherry picked from commit 4050ddc59beeff2be5a862782579ceb8e5775c60)
12. Commit 4050ddc59beeff2be5a862782579ceb8e5775c60 in lucene-solr's branch refs/heads/master from Chris M. Hostetter [https://gitbox.apache.org/repos/asf?p=lucene-solr.git;h=4050ddc] Harden RulesTest * ensure all collections/replicas are active * use waitForState or waitForActiveCollection before checking rules/snitch to prevent false failures on stale state * ensure cluster policy is cleared after each test method Some of these changes should also help ensure we don't get (more) spurious failures due to SOLR-13616