Item 330
git_comments:

1. add a few more docs
2. kill non leader - new leader could have all the docs or be missing one
3. for(int i = 0 ; i < nodesDown.size(); i++) { bringUpDeadNodeAndEnsureNoReplication(shardToLeaderJetty.get("shard1"), neverLeader, false); } checkShardConsistency(false, true);
4. index enough docs and commit to establish frame of reference for PeerSync
5. node failure and recovery via PeerSync
6. slow down adds, to get documents indexed while in PeerSync
7. make sure leader has not changed after bringing initial leader back
8. first shutdown a node that will never be a leader
9. two docs need to be sync'd back when replica restarts
10. now shutdown all other nodes except for 'nodeShutDownForFailure'
11. bring back dead node and ensure it recovers
12. skip the randoms - they can deadlock...
13. tlog gets deleted after node restarts if we use CachingDirectoryFactory. make sure that tlog stays intact after we restart a node
14. random non leader node
15. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
16. now shutdown the original leader
17. bring back all the nodes including initial leader (commented as reports Maximum concurrent create/delete watches above limit violation and reports thread leaks)
18. disable fingerprint check if needed
19. sleep for a while for leader to change ...
20. bring up node that was down all along, and let it PeerSync from the node that was forced to PeerSynce
21. * * Test sync peer sync when a node restarts and documents are indexed when node was down. * * This test is modeled after SyncSliceTest
22. Throwing an error here will kill the thread
23. find fingerprint for max version for which updates are requested
24. overwrite fingerprint we saved in 'handleVersions()'
25. check our fingerprint only upto the max version in the other fingerprint. Otherwise for missed updates (look at missed update test in PeerSyncTest) ourFingerprint won't match with otherFingerprint
26. fingerprint should really be requested only for the maxversion we are requesting updates for In case updates are coming in while node is coming up after restart, node would have already buffered some of the updates. fingerprint we requested with versions would reflect versions in our buffer as well and will definitely cause a mismatch

git_commits:

1. **summary:** SOLR-9310: PeerSync fails on a node restart due to IndexFingerPrint mismatch
   message: SOLR-9310: PeerSync fails on a node restart due to IndexFingerPrint mismatch

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** PeerSync fails on a node restart due to IndexFingerPrint mismatch
   description: I found that Peer Sync fails if a node restarts and documents were indexed while node was down. IndexFingerPrint check fails after recovering node applies updates. This happens only when node restarts and not if node just misses updates due reason other than it being down. Please check attached patch for the test.

jira_issues_comments:

1. [~yseeley@gmail.com] and [~markrmiller@gmail.com], in [SOLR-8690|https://issues.apache.org/jira/browse/SOLR-8690] you mentioned that fingerprint check could have performance cost. Was performance cost you mentioned could be due to the fact that PeerSync was failing on node restart and hence Solr was falling back to do full replication ? Is PeerSync failing on node restart expected behavior
2. Adding [~k317h] in the loop
3. I was able to figure a few more things about what going own. The existing {{PeerSyncTest}} does not change core's state from {{ACTIVE}} to recovering and hence the condition following condition (block) in {{DistributedUpdateLogProcessor.versionAdd()}} does not get executed {code} if (ulog.getState() != UpdateLog.State.ACTIVE && (cmd.getFlags() & UpdateCommand.REPLAY) == 0) { // we're not in an active state, and this update isn't from a replay, so buffer it. cmd.setFlags(cmd.getFlags() | UpdateCommand.BUFFERING); ulog.add(cmd); return true; } {code} However, the test I have attached certainly goes through above mentioned code which results in updates came from log replay to be buffered. As a result of this {{compareFingerPrint()}} check at the end of {{PeerSync.handleUpdates()}} fails,whenever a PeerSync was triggered and core was in not ACTIVE state. I am not entirely sure why a core would in ACTIVE state if PeerSync was triggered (which happens in {{PeerSyncTest}}). I think {{compareFingerPrint}} check should be moved out of {{PeerSync}} class to {{RecoveryStrategy}} after buffered updates are applied. It might also be a good idea to move the commit after log replay as current commit seems to be resulting in a NOOP.
4. Adding [~andyetitmoves] in the loop
5. On second thought, moving fingerprint check after log replay would not work as it can contain updates other than the node received from the leader. Only two options I could think of is * Change the check in {{DistributedUpdateProcessor.versionAdd()}} to handle {{PEER_SYNC}} flag (i.e. not to buffer updates from PEER_SYNC). I am not entirely sure what other issues it might cause. * Not to do the fingerprint check for after updates are applied in PeerSync if core is not in active state.
6. Attached patch to fix the issue. Fix is not to buffer updates with {{UpdateCommand.PEER_SYNC}} in {{DistributedUpdateProcessor.addVersion()}}
7. If fix itself looks good, I will upate my Test to include all the scenarios in exiting {{PerrSyncTest}} test
8. Although my patch would work, if there was no active indexing going on during PeerSync, fingerprint check may fail if there was any active indexing was going on. There are too many race conditions here. In my opinion people who are continuously indexing data, should disable fingerprint check.
9. There is another problem in {{RealTimeGetComponent.processGetVersions()}}, since asking for a fingerprint causes a new RealTime Searcher to open, we should first get the fingerprint and then get versions from ulog
10. Updated patch and tests for scenarios I have described
11. is there a test in the patch? I don't see it
12. [~noble.paul] - Thanks for pointing out. Check updated patch with the test
13. When you post a patch please post the whole patch, not pieces. This contains the test only. We can't know in which order to apply your patches. No need to delete old patches.
14. Damn the first lapse was 4:00 am mystery thing, this second on is on me. Recomputing and attaching patch.
15. I'm assuming that this one is not related to this ticket. Should it not be a new ticket?
16. I think it is related to this ticket. It is just another scenario in which PeerSync would fail on restart "if you index documents while a node was down, but did not issue a commit (or not autoCommit / autoSoftCommit was triggered). Do you think there needs to be a different ticket for this scenario. If needed, I will create a sperate ticket, update my patch for this ticket not to cover that scenario and send a patch for this scenario on the new ticket

17. I guess we should compare the fingerprint before the updates are applied. I'm testing another approach. Will report back
18. That won't work as leader may have already diverged when compared to node that is just coming up
19. Well, There is a way to compute a fingerprint upto a point . So, if replica has updates till version 'x' and if fingerprint of the leader matches till 'x' wouldn't it be good enough?
20. bq. The variable name is confusing. maxInHash is not a hash max-in-hash... t's the maximum value included in the hash. That name does not seem to imply that it *is* a hash.
21. Another question. Why update commands with flag {{REPLAY}} are not buffered in {{DistributedUpdateProcessor.versionAdd()}} but update commands with flag {{PEER_SYNC}} are. Also even you ask for IndexFingerPrint of a specific version, you will end up sending {{Long.MAX_VALUE}} while asking for versions to {{RealTimeGetComponent}}. Given all different race conditions, I am not sure how much we really gain out of fingerprint check
22. [~praste] I have tried a few approaches and only your approach (as given in the patch) seems to yield results. I'll do some more testing anyway. Your test has some commented out stuff, parallel indexing. Does it work too?
23. Parallel indexing unfortunately does not work, it is a race condition where things could diverge on the leader during call to get index fingerprint and get versions. I think fingerprint check is mostly likely to fail if documents are being indexed while node is recovering and if a commits (or softCommits) get issued/triggered.
24. Another approach. Compute the fingerprint with the latest version in the current node and compare it with the same version in the remote node
25. Looks like you uploaded wrong patch. It doesn't look any different than one I attached.
26. sorry , wrong file
27. Thanks [~rideordie482@gmail.com], I will take a look at it tonight and run my origin test that exposed the issue and will let you know if it works as expected. Does changing logic in the {[DistributedUpdateProcessor}} has any downside to it? I am wondering why {{REPLAY}} flag is treated differently than {{PEER_SYNC}}
28. [~noble.paul] - Patch looks good. Can you please provide info about my question for {{PEER_SYNC}} vs {{REPLAY}} flag. are there any downsides of the way I was doing it? Only problem I could think of your approach is we are requesting updates twice, which in my case is asking for tens of thousands of updates, which could be lot of chatter of the wire.
29. bq. am wondering why REPLAY flag is treated differently than PEER_SYNC REPLAY updates are not written to update log because that would be redundant. The update log itself is the source of the updates being replayed. PEER_SYNC updates come from a different node and they certainly weren't present in the local update log already (or else we would not have requested them). This is why they must be written to the tlog.
30. [~noble.paul] - Seems like with your patch we are matching fingerprint upto the version before node went down, whereas intent is to compare fingerprint after apply updates from the leader. I modified {{PeerSync.handleUpdates()}} to not apply updates at all and fingerprint check still passed. Here is excerpt of change I am talking about {code} private boolean handleUpdates(ShardResponse srsp) { // we retrieved the last N updates from the replica boolean test = true; List<Object> updates = (List<Object>)srsp.getSolrResponse().getResponse().get("updates"); SyncShardRequest sreq = (SyncShardRequest) srsp.getShardRequest(); if(test) { return compareFingerprint(sreq); } ... {code} I am not sure if there is a way to write a test for my observation about fingerprint check happening on state before updates were applied and not after were applied. This probably defies intent of comparing fingerprint after applying updates. Why not check it before asking for updates in the first place then.
31. [~noble.paul] just wanted to check on the current state here. Do you suggest holding back the 5.5.3 release for this ? Please consider the complexity as well as how close we are to the solution at this point.
32. it depends. if there is no urgency, let's hold back 5.5.3. Else there can always be a 5.5.4
33. bq.This probably defies intent of comparing fingerprint after applying updates. Why not check it before asking for updates in the first place then. Isn't it enough if we compare the fingerprint up to the point when it was down? After that, we are going to apply the updates from the other replica (so fingerprint of the delta will be same anyway. If we can get a consensus on this approach, I guess we should implement this solution
34. What was the original reason for adding fingerprint check. Unless that is clear we are not solving the right problem here. I couldn't find a test case that led to adding fingerprint checl. * If intent is to check state before node down, for clarity. check should be made before applying updates. * Let's also make read and sort update log of recovering node only once. * I would also uncomment parallel indexing logic in the {{PeerSyncReplicationTest}}, as this won't break fingerprint check, with your patch.

35. bq.What was the original reason for adding fingerprint check? Previously we were just comparing the latest versions. We had no way to know if some intermediate versions were missing or not. In most cases, that would be correct. but if there was an out of order update, then we would assume we have everything and go ahead with downloading versions after our latest update. Fingerprints compute the hash of all versions. So , we will be able to avoid such errors bq.Let's also make read and sort update log of recovering node only once. Sure. I'm aware of this optimization. It was a rudimentary patch to ensure that the approach is valid
36. bq.Here is excerpt of change I am talking about Somehow I could not make that change to work. Can u post your patch
37. This an experimental patch, just to test what happens if PeerSync falls on its face. I could not think of an elegant way to test it, but it just proves that we if compare index fingerprint only before applying updates, it would not validate if PeerSync itself is successful. IMHO we should know that after recovery nodes are in sync irrespective of whether they were in sync at some point before failure or not.
38. Also looking at Yonik's comments https://issues.apache.org/jira/browse/SOLR-8586?focusedCommentId=15122263&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-15122263 https://issues.apache.org/jira/browse/SOLR-8586?focusedCommentId=15126352&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-15126352 It seems to me that we should make sure that nodes are in sync after applying updates. I had tagged Yonik and Mark in my comments but haven't heard anything from them yet. May be we should reach out to them to ask what's the intention of fingerprint check.
39. It's hard to have the fingerprint match after applying the updates. The system as exists today does not work in case of restarting node or LIR. So , it's already broken. I shall wait for some time for [~yonik@apache.org] to revert on this.
40. Yeah, it was better before. Except for the fact that it didn't work.
41. I Plan to commit this soon if there are no other comments
42. Can this wait for Yonik's comment. Je fact his implementation used MAX VERSION and that check was added at the end of handleUpdates makes me think intention was to do the check later and not before starting PeerSync
43. bq. intention was to do the check later and not before starting PeerSync Yes, that was the intention. The approach was deliberately conservative : in cases where peersync formerly thought it had gotten into sync, verify that with a fingerprint. On a quick skim through, it's not clear what people think the bug is... but I'll try a more thorough read-through next.
44. The testcase fails without the fix. That can help you see the bug easily
45. You were ready to commit a fix though... so what was exactly is the bug, what caused it, and how does the patch fix it?
46. Here is short description of bug 1. A node goes down in solr cloud 2. More documents and added (and may be a commit issued) 3. Node that was down comes up. 4. Node gets fingerprint from the leader and version too 5. Node calculates diff for missing versions and requests updates for the same 6. Node applies updates and then checks it's fingerprint against the leader's fingerprint 7. Check in #6 always fail, fingerprint of recovering node does not reflect updates applied during PeerSync There are two proposed fixes * My fix is not to buffer updates commands that have PEER_SYNC flag on it. I think, hesitation about my patch, we don't know what other side effect it may have. (All test cases are passing, but we might not have a test case where my fix would break things) * Noble's fix to check fingerprint before we start applying updates. In my opinion this no really fixing original issue, what really matters is if fingerprint matches after applying updates during PeerSync. and we don't know which approach in right. May be there 3rd better approach
47. bq. My fix is not to buffer updates commands that have PEER_SYNC flag on it. Ah, I see... I had no idea that PEER_SYNC updates were buffered (I don't recall that being intentional at least). That would prevent updates obtained from the leader and applied to the replica from being included in the fingerprint on the replica after the peersync.
48. bq. I think, hesitation about my patch, we don't know what other side effect it may have We know the side effect. If this replica becomes the leader and some other node tries to peersync from this replica, that node will not get those updates. That breaks the functionality
49. [~noble.paul] - I will write a test for the scenario you mentioned
50. Hesitation about your patch is it checks index fingerprint before applying updates. Which Yonik agrees is not right.
51. If there is a way to get the fingerprint after applying the updates, it will be good.
52. Do you see a potential problem if fingerprint is compared before applying updates
53. If something goes wrong while applying updates, how can you assure that replica is in sync?

54. Of something goes wrong peersync fails
55. Here is my patch with update test for scenario Noble is concerned about and a small diagram depicting what is going on in the test. Test for my patch still passes. If it looks good, I will take stab at fixing PeerSync failure (due index fingerprint) during active indexing, however, due to race conditions, it might not work well. Can we setup a call to discuss this ?
56. Final patch. Here are highlights about the changes * Don't buffer updates with `PEER_SYNC` flag on it, otherwise those would not be included in the fingerprint and PeerSync would always fail on fingerprint check * PeerSync should care for fingerprint only for the updates it is applying. Otherwise if documents are being indexed while a node is PeerSync. Node recovering would have some of the documents in a buffer already. Node recovering would ask for updates for those documents. Leader's fingerprint will however reflect all the documents but recovering node's fingerprint would not consider documents in the buffer. This also applies for missed updates * Cache fingerprint in `SolrIndexSearcher` judiciously
57. Use the maxVersion in fingerprint before returning versions
58. Since any race condition would be handled by limiting fingerprint upto the maxVersion will get in `getUpdates`, I could not think of compelling reason to uset in `getVersions()`. It probably won't hurt to use `maxVersion` before returning versions
59. bq.It probably won't hurt to use `maxVersion` before returning versions You are right. It won't hurt peersync because getUpdates take care of it. But , for correctness of the API the the following should be true {{fingerPrint.maxEncounteredVersion== max(versions)}}
60. Commit c2e769450fac21a8f98e818b4783d7dca14cffb8 in lucene-solr's branch refs/heads/master from [~noble.paul] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=c2e7694] SOLR-9310: PeerSync fails on a node restart due to IndexFingerPrint mismatch
61. Commit 80f916780798162a5c68875fed10ef1ff132c8f7 in lucene-solr's branch refs/heads/master from [~noble.paul] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=80f9167] SOLR-9310: PeerSync fails on a node restart due to IndexFingerPrint mismatch, precommit errors fixed
62. Commit 2f73129edae7541d5cf45c2085d9ca40ff048b9b in lucene-solr's branch refs/heads/branch_6x from [~noble.paul] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=2f73129] SOLR-9310: PeerSync fails on a node restart due to IndexFingerPrint mismatch
63. Commit c37c22dbb0cd6de5804b1d72c4e2e86c1bba7ef2 in lucene-solr's branch refs/heads/branch_6x from [~noble.paul] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=c37c22d] SOLR-9310: PeerSync fails on a node restart due to IndexFingerPrint mismatch
64. Reopening... the cache that was added in SolrIndexSearcher is not thread safe and is checked outside synchronization. Also, about this comment: {code} // TODO what happens if updates came out of order, would cached fingerprint still be valid? // May be caching fingerprint may lead more problems {code} The index fingerprint only depends on what documents are in the index, not on their order in the index. And since the cache is on SolrIndexSearcher (which has a static view of the index), it will be impossible for fingerprint to change for a given max version. The comment should probably just be removed to avoid confusion. In changes on UpdateLog: {code} + if(ptr.version > maxVersion) continue; {code} versions can be negative for deletes, so we should really be checking against the absolute value of ptr.version
65. What is the need for the cache? I seems like there would only ever be a cache hit if if there is no active indexing. It seems like the added complexity is not worth the potential small performance boost.
66. bq. What is the need for the cache? Multiple replicas syncing against each other (or all replicas syncing against a new leader...) I *think* it's part of the protocol to elect a new leader (because when one leader goes down, we don't know which replicas may have received the last update(s) and which replicas did not...) In such a scenario, there is no active indexing because no new leader yet. Some people run with large numbers of replicas (10 or 20), and hence the difference could well be large.
67. Addressing the thread safety issue and compare absolute values of versions in UpdateLog
68. minor: since there is a cache now, we could sync on that and get rid of fingerprintLock
69. 👍
70. Commit 37ae065591772172dbd44cde4c952d7b56fc8803 in lucene-solr's branch refs/heads/master from [~noble.paul] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=37ae065] SOLR-9310: fixing concurrency issue and taking care of negative versions
71. I see the double-check locking pattern there... I believe ConcurrentHashMap.computeIfAbsent would be perfect in this case?
72. I tried that, but since `IndexFingerprint.getFingerprint()` can throw an exception, implementation using `computeIfAbsent` looked too ugly to use it.
73. Commit 0bdbbbfd52a95e83ce3827161852dbccdd618f5b in lucene-solr's branch refs/heads/branch_6x from [~noble.paul] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=0bdbbbf] SOLR-9310: fixing concurrency issue and taking care of negative versions

74. ported to 5.5.x
75. Commit c6c3166bf5d28922bb3639ac9da3912aab85f520 in lucene-solr's branch refs/heads/master from [~noble.paul] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=c6c3166] SOLR-9310: addressing the test failures in jenkins
76. Commit d9c4c5282a46bc5d3d1a6e4b1586083dc8970837 in lucene-solr's branch refs/heads/branch_6x from [~noble.paul] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=d9c4c52] SOLR-9310: addressing the test failures in jenkins
77. Ah, a piece of the puzzle: we didn't always buffer before peersync... SOLR-8407
78. Thanks Yonik. I don't know context for SOLR-8407. Should we have not buffered updates to fix SOLR-8407, or patch for this (SOLR-9310) issue would break SOLR-8407
79. Mark pointed me to SOLR-8085 for where/when we started buffering before peersync. I think it probably had to do with the following scenario: 1) replica that is behind comes up and starts peersync 2) replica receives a bunch of updates forwarded from leader (and indexes them and adds them to tlog as normal) 3) replica goes down 4) replica comes up, looks at last 100 versions in it's transaction log, does peersync and concludes that it's up-to-date Putting the udpate log in buffering mode adds a FLAG_GAP to all the records int he log, which signals that there is a gap somewhere and hence one can't conclude that if the last 100 updates are good that everything else is good. This is all before fingerprinting of course.
80. Commit 8655b97b27d8da470c8235683af11a8b85a2b10f in lucene-solr's branch refs/heads/branch_5_5 from [~noble.paul] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=8655b97] SOLR-9310: java 7 compile errors
81. Commit 4f6e2546739e5352738f786aaddfb6f08b1549aa in lucene-solr's branch refs/heads/branch_5x from [~noble.paul] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=4f6e254] SOLR-9310: PeerSync fails on a node restart due to IndexFingerPrint mismatch
82. Commit afcc8c05dd2c13a0cb0165674931a99decf98373 in lucene-solr's branch refs/heads/branch_5x from [~noble.paul] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=afcc8c0] SOLR-9310: PeerSync fails on a node restart due to IndexFingerPrint mismatch
83. I've manually checked that these changes are in branch_5_5, but there's no log/comment here in the JIRA, other than the single entry about fixing Java7 compile errors. Am I missing something here?
84. Actually, the commit did not make it to JIRA. May be the bot was down or something.
85. I looped HdfsChaosMoneyNothingIsSafeTest after this change, and I occasionally started getting some shards "non consistent" failures again.
86. Is this happening when active indxing is going on when node is in recovery. That is only scenario I could think of fingerprint might mismatch (but I don't really doubt that), as we compare fingerprint only upto the max version that we request during peersync. Also, earlier, the only reason nodes were able to sync was nodes were doing snap pull every single time. Does this fail only with HDFS or in non-hdfs scenario as well. If this breaks only with HDFS, may be we can add option to recover using replication only until the fingerprint implementation stabilizes.
87. I only tested with the HDFS variant quickly... it tends to fail a little more often because of greater timing variability I think. We should also test the "safe-leader" test... this may be just an issue when leaders are killed.
88. This wasn't backported to 6.2.1 so removing the fix version 6.2.1
89. I just happened to notice that PeerSyncReplicationTest failed in jenkins: https://jenkins.thetaphi.de/job/Lucene-Solr-6.x-Linux/1796/ Not sure how often it's happened though.
90. I went through logs at https://jenkins.thetaphi.de/job/Lucene-Solr-6.x-MacOSX/429/consoleFull If PeerSync was unsuccessful I would expect to see a line like {{o.a.s.u.PeerSync Fingerprint comparison: -1}} However, I don't see such line. I could think of two scenarios that could break the test * data directory could get deleted while a node is brought down, since data directory is created in {{temp}}. Upon restart replica would have no frame of reference and will have to fall back on replication. * we need a better check than relying number of requests made to {{ReplicationHandler}}
91. Closing after 6.3.0 release.