Item 86
git_comments:

git_commits:

1. **summary:** SOLR-5476 removing forbidden API usage
   message: SOLR-5476 removing forbidden API usage git-svn-id: https://svn.apache.org/repos/asf/lucene/dev/trunk@1558846 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Overseer Role for nodes
   description: In a very large cluster the Overseer is likely to be overloaded.If the same node is a serving a few other shards it can lead to OverSeer getting slowed down due to GC pauses , or simply too much of work . If the cluster is really large , it is possible to dedicate high end h/w for OverSeers It works as a new collection admin command command=addrole&role=overseer&node=192.168.1.5:8983_solr This results in the creation of a entry in the /roles.json in ZK which would look like the following {code:javascript} { "overseer" : ["192.168.1.5:8983_solr"] } {code} If a node is designated for overseer it gets preference over others when overseer election takes place. If no designated servers are available another random node would become the Overseer. Later on, if one of the designated nodes are brought up ,it would take over the Overseer role from the current Overseer to become the Overseer of the system

jira_issues_comments:

1. The strategy to implement this is as follows Prioritization logic run at Overseer node * Any Overseer elect would first check for the presence of /roles.json in ZK. If not return * lookup the children of /overseer_elect node and see if there are nodes with Overseer roles among children. If not return * All nodes whose seq number is smaller than that of the first Overseer role node is asked to disconnect and reconnect . So that those nodes will append themselves to the tail of the list and the Overseer role node would come up at the head * After this process , the Overseer checks if it is itself an Overseer role node , If not ,relinquish it's Overseer position and go back to election If a new node comes up who is assigned an Overseer role, it has to ensure that it is at the head of the election queue , * If any new node comes up, it checks for the /roles.json . If present , checks if it is assigned an overseer role, if yes , it checks if the head of the election queue is a node with Overseer role. If not, send a message to the current Overseer to run the prioritization logic
2. New command implemented addrole and removerole . The only supported role now is overseer
3. added a testcase
4. some errors fixed
5. **body:** patch with more logging
   label: code-design
6. **body:** code made more robust
   label: code-design
7. Commit 1558760 from [~noble.paul] in branch 'dev/trunk' [https://svn.apache.org/r1558760] SOLR-5476 Overseer Role for nodes
8. Commit 1558776 from [~noble.paul] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1558776] SOLR-5476 Overseer Role for nodes
9. Commit 1558829 from [~noble.paul] in branch 'dev/trunk' [https://svn.apache.org/r1558829] SOLR-5476 , logging added

10. Commit 1558830 from [~noble.paul] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1558830] SOLR-5476 , logging added
11. Commit 1558846 from [~noble.paul] in branch 'dev/trunk' [https://svn.apache.org/r1558846] SOLR-5476 removing forbidden API usage
12. Commit 1558847 from [~noble.paul] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1558847] SOLR-5476 removing forbidden API usage
13. Commit 1559043 from [~thetaphi] in branch 'dev/trunk' [https://svn.apache.org/r1559043] SOLR-5476: Fix forbidden. PLEASE RUN "ant precommit" (root) or alternatively the faster "ant check-forbidden-apis" (in your module folder) before committing!
14. Commit 1559044 from [~thetaphi] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1559044] Merged revision(s) 1559043 from lucene/dev/trunk: SOLR-5476: Fix forbidden. PLEASE RUN "ant precommit" (root) or alternatively the faster "ant check-forbidden-apis" (in your module folder) before committing!
15. Commit 1559100 from [~noble.paul] in branch 'dev/trunk' [https://svn.apache.org/r1559100] SOLR-5476 logging added
16. Have you actually run into issues with this? Seems like premature optimization...the overseer simply fires http commands and simple zk stuff...you really think 'special hardware' overseers are going to matter and not just complicate the code?
17. **body:** It is not really about special H/W , but mostly about 'dedicated' H/W. If the same Overseer node is used for normal cores there is a good chance that those nodes do some CPU intensive operations or GC pauses which would delay Overseer operations. This can cause piling up of messages in large clusters
    label: code-design
18. Commit 1559399 from [~noble.paul] in branch 'dev/trunk' [https://svn.apache.org/r1559399] SOLR-5476 ftests are being fixed . SO ignore for the time being
19. Commit 1559402 from [~noble.paul] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1559402] SOLR-5476 ftests are being fixed . SO ignore for the time being
20. Commit 1559405 from [~noble.paul] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1559405] SOLR-5476 ftests are being fixed . SO ignore for the time being
21. Commit 1559677 from [~noble.paul] in branch 'dev/trunk' [https://svn.apache.org/r1559677] SOLR-5476 tests passing
22. Commit 1559680 from [~noble.paul] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1559680] SOLR-5476 tests passing
23. Commit 1559936 from [~noble.paul] in branch 'dev/trunk' [https://svn.apache.org/r1559936] SOLR-5476 handle nonode exception
24. Commit 1559937 from [~noble.paul] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1559937] SOLR-5476 handle nonode exception
25. This test is still failing on almost all of my local test runs.
26. Commit 1560279 from [~noble.paul] in branch 'dev/trunk' [https://svn.apache.org/r1560279] SOLR-5476 hardened a bit
27. Commit 1560280 from [~noble.paul] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1560280] SOLR-5476 hardened a bit
28. [~markrmiller@gmail.com] What is the sure shot way for Overseer/OverseerCollectionProcessor to quit and somebody else take its place? The failures are due to the leader not giving up
29. Perhaps you can create your own instance of overseer and force election just like OverseerTest does with OverseerRestarter?
30. Commit 1560314 from [~noble.paul] in branch 'dev/trunk' [https://svn.apache.org/r1560314] SOLR-5476 ignored till there is a reliable way to restart Overseer
31. Commit 1560315 from [~noble.paul] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1560315] SOLR-5476 needs a reliable way to restart Overseer
32. Commit 1562434 from [~noble.paul] in branch 'dev/trunk' [https://svn.apache.org/r1562434] SOLR-5476 tests were failing earlier
33. Commit 1562435 from [~noble.paul] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1562435] SOLR-5476 tests were failing earlier
34. See common jenkins test fail - something is off here, with the impl or the test. An Overseer reading the queue can run into another Overseer having already removed the distrib queue node being looked at.
35. I don't think this is right. Where do you ensure the old Overseer is stopped first? You can't just force a new one by deleting the election node and doing the run leader process.
36. **body:** Trying to force a leader like that is complicated, and I think there are a bunch of holes here.
    label: code-design

37. **body:** I think the tests are too light as well. You want strong testing around this special overseer takeover - there are so many things that could go wrong - you need to make sure things continue merrily after several disaster scenario. And it needs to ensure that there are never more than one Overseer running at a time. I think this was committed before it was ready.
    label: test
38. **body:** I've worked a little to try and do some of the right things along the current approach, but I'm starting to feel I'm -1 on this approach to leader prioritization. I think if there was proper testing here, no way doing things this way would survive it, and I'm not sure I can make this idea work. A better first approach probably involves having prospective leaders give up there position in line if they see a better candidate behind them. The trick then being to look out for how to deal with a node that looks like a better candidate but for some reason keeps failing to be leader.
    label: code-design
39. **body:** bq.A better first approach probably involves having prospective leaders give up there position in line if they see a better candidate behind them. That can also be done too. But it is generally better to keep the Overseer designates line up first and second (if there are more than one) in the queue to avoid delay and it is easy to do also. Usually the command is invoked to assign an overseer after the Overseer is already elected and running. In that scenario we need to force the current Overseer to give up their position .
    label: code-design
40. **body:** The problem is the reliance on the isLeader checks in the over seer threads to kill the previous overseer. It's a race, and the tests are showing how easy it is for multiple overseers to interfere with each other. We have to avoid those races, both because the current impls require it, but also because who knows what future things the overseer may do. The Overseer must be a mutually exclusive process 100%. I'm not sure that we can bullet proof this approach. Perhaps. But I wouldn't believe it until we had some much stronger testing. The current test fails almost every run for me when I run it in eclipse because Overseers interfere with each other. I'm also a little scared of the way this messes with the election process, but that appears like it should be okay.
    label: test
41. The problem of race condition happens on the first time ADDROLE is invoked . Once you have added a couple of Overseers designate it won't happen again because the Overseer designates are always in the front for election and then there is no need to kill a running overseer
42. It's a huge problem. As you can see from the extremely common test fails. Currently, you see a fail were it kills the processing thread. That race is a big deal. I'm -1 on this for 4.7 given the problems with it. Something has to be done that doesn't compromise the Overseer running mutually exclusively. This is important now and for the future. We need to ensure the Overseer cannot possibly step on itself, and this feature pretty much ensures that it will. I'm -1 on the feature until the implementation is fixed.
43. What is the best strategy to kill an existing overseer? Once we agree upon that it is easy to fix it
44. The test does not fail . It errors out . Because I forcefully kill the overseer.
45. This is to avoid the race condition In this patch * The new overseer is asked to wait for the STATE_UPDATE_DELAY before processing the queue items * STATE_UPDATE_DELAY is the maximum interval between the Overseer checks for amILeader().
46. I don't know that that is a great solution either though. For one, future overseer tasks may have nothing to do with the state delay. For two, you don't know how long any overseer job will take - you might wait that delay, but the overseer may be taking 30 seconds or more to run a command and then when it done it removes a queue item out from the new overseer, killing the thread. I can't work out how this protects against Overseers completely from overlapping. I've thought through a lot of this type of thing, and it just seems like there are lots of holes that are very hard to fill on this path.
47. **body:** The in-flight operations for Overseer are really short running. Keeping the delay slightly longer than STATE_UPDATE_DELAY would not hurt. Again , the problem of Overlap because of long running operations is no just restricted to the Overseer Roles feature it can happen otherwise too.
    label: code-design
48. bq. Again , the problem of Overlap because of long running operations is no just restricted to the Overseer Roles feature it can happen otherwise too. It's not likely to happen otherwise. Those checks that you are now using like a feature are really just fail safes. Emergency shutoffs to minimize damage in a bad situation. They are not meant to be hit. Normally an Overseer will be an Overseer until it dies, crashes, cases to exist. There will be no overlap in any normal circumstances. This patch changes that.
49. Commit 1566247 from [~noble.paul] in branch 'dev/trunk' [https://svn.apache.org/r1566247] SOLR-5476 avoiding race condition

50. Commit 1566248 from [~noble.paul] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1566248] SOLR-5476 avoiding race condition
51. I just saw this fail again locally with the latest commit.
52. I ran it around 20 times on my local box and could not get an error/failure . I'll look at jenkins for failures
53. I've seen it fail a couple times already - something about failing to reorder.
54. To sum up an offline conversation Noble and I had about this: I'm not happy with the approach, I think it's dangerous, and I'm worried about making more and more of these types of exceptions. However, I will withdraw my -1 on it if the testing is significantly beefed up and it can be proved the current approach is usually not going to screw you. I think the current test is an absolute minimum test and it has regularly failed on jenkins (and for me locally even more fequently) since it was committed. We also agreed on a JIRA issue to come up with a better approach at some point. I also think we need to add some defensive programming. If we try and remove a queue item that is not there, we should assume another overseer already ate it rather than letting the Overseer thread die.
55. Another testcase to kill an existing overseer and checking for another one to takeover
56. Commit 1566620 from [~noble.paul] in branch 'dev/trunk' [https://svn.apache.org/r1566620] SOLR-5476 added testcase
57. Commit 1566624 from [~noble.paul] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1566624] SOLR-5476 added testcase
58. Commit 1567012 from [~noble.paul] in branch 'dev/trunk' [https://svn.apache.org/r1567012] SOLR-5476 added testcase
59. Will this feature lend itself to dynamically updating the role of a node -- so depending on the state of the environment, I can grant or revoke the Overseer role? I guess the collection admin will technically allow for it, but just wanted to check if that's something that can be routinely done when the cluster is live without any performance or stability impact.
60. **body:** Yes, you can invoke the collection API on a live system. The performance impact will be there for overseer operations for a few seconds . Regular read /write should not be impacted at all
    label: code-design
61. **body:** The testing for this is still super minimal. Also the following was not addressed: {noformat} I also think we need to add some defensive programming. If we try and remove a queue item that is not there, we should assume another overseer already ate it rather than letting the Overseer thread die. {noformat} I don't think much of what I was concerned about was addressed. The test can also still fail because two Overseers are running at the same time (Mike just saw this). If this didn't go out in 4.7 I would bring back the -1. This is why I said on the phone that I would prefer we mark it as experimental - so that it could still be vetoed after release if the problems were not addressed.
    label: code-design
62. **body:** bq.I also think we need to add some defensive programming. If we try and remove a queue item that is not there, we should assume another overseer already ate it rather than letting the Overseer thread die. We are trying to kill an overseer safely. What is being done is wait for STATE_UPDATE_DELAY+100 ms and then start the next Overseer . I agree that this is not 100% foolproof (but I have not seen OverseerRolesTest fail for quite sometime) We can definitely add more checks to ensure a proper handing over of overseer role. I didn't quite get the above suggestion clearly. Please elaborate . I'll be glad to implement that I think this should be an issue in itself rather than confusing with the "Roles" issue. bq.This is why I said on the phone that I would prefer we mark it as experimental Where do I need to sign? BTW , I would say the users of this feature are very unlikely to hit this issue
    label: code-design
63. I'll open another issue for Overseer restart and I will work on hardening that. This probably is not the right place