Item 75
git_comments:

1. * 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.
2. * * Returns the data at the first element of the queue, or null if the queue is * empty after wait ms. * * @param wait max wait time in ms. * @return data at the first element of the queue, or null.
3. leader changed - close the overseer
4. if an event comes in the next 100ms batch it together
5. TODO: we need to think carefully about what happens when it was a leader that was expired - as well as what to do about leaders/overseers
6. this is troublesome - we dont want to kill anything the old leader accepted though I guess sync will likely get those updates back? But only if ExecutorUtil.shutdownAndAwaitTermination(cc.getCmdDistribExecutor()); Overseer.createClientNodes(zkClient, getNodeName());

git_commits:

1. **summary:** SOLR-5436: Eliminate the 1500ms wait in overseer loop as well as polling the ZK distributed queue.
   message: SOLR-5436: Eliminate the 1500ms wait in overseer loop as well as polling the ZK distributed queue. git-svn-id: https://svn.apache.org/repos/asf/lucene/dev/branches/branch_4x@1544257 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Eliminate the 1500ms wait in overseer loop
   description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
2. **summary:** Eliminate the 1500ms wait in overseer loop
   description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
3. **summary:** Eliminate the 1500ms wait in overseer loop
   description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
4. **summary:** Eliminate the 1500ms wait in overseer loop
   description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
5. **summary:** Eliminate the 1500ms wait in overseer loop

description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor

6. **summary:** Eliminate the 1500ms wait in overseer loop
   description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
7. **summary:** Eliminate the 1500ms wait in overseer loop
   description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
8. **summary:** Eliminate the 1500ms wait in overseer loop
   description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
9. **summary:** Eliminate the 1500ms wait in overseer loop
   description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
10. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
11. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
12. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
13. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
    label: code-design
14. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
15. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
16. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
17. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
18. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
19. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in

OverseerCollectionProcessor

20. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
21. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
22. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
23. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
24. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
25. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
26. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
    label: code-design
27. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
    label: code-design
28. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
29. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
30. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
31. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
32. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
33. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor

34. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor
35. **summary:** Eliminate the 1500ms wait in overseer loop
    description: The Overseer thread waits 1500 ms before it polls for new events. The wait should be eliminated and it should just wait for new events till they come the way it is done in OverseerCollectionProcessor

jira_issues_comments:

1. As I understand it, the reason behind the wait are to group the processing of multiple events together so that the cluster state change is not broadcasted to all watching nodes too many times. But this can work along with splitting the cluster state in SOLR-5381.
2. [~shalin] Originally we did not have a blocking lookup of the queue. that was also one of the reasons.
3. bq. Originally we did not have a blocking lookup of the queue Right - the history is that Sami wrote the Overseer loop first, later I wrote the OverseerCollectionProcessor loop - I didn't like the polling and used the blocking lookup of the queue and just never got to catching the Overseer loop impl up.
4. What if there are state change events happening in quick succession (like a rolling restart) will it not lead to too many updates getting propagated to all the nodes? Is it a real problem?
5. Shalin's original comment? Yeah, I suppose it does make sense. How much of a problem it is probably depends on the scale.
6. In my patch it would ensure that any event would get processed within the STATE_UPDATE_DELAY or all normal events would get processed within 100ms
7. added a new method to peek(waitTime) to DistributedQueue
8. This time w/ all the tests passing [~markrmiller@gmail.com] please do a review . If it is fine I can check it in soon
9. Yeah, I'll take a look today.
10. I've looked it over, but I have to look closer at a couple things and I got tied up in other issues. Let me take a closer look tomorrow.
11. optimized the DistributedQueue code
12. I'm reviewing the latest patch.
13. **body:** Looks good. Here is the same patch but with some minor cleanup and that eliminates the 5 second poll of the main queue.
    label: code-design
14. I kept the 5sec peek() on purpose. Because if the OverSeer changes and no state event happens there is no way for the node to know it. So let's keep it
15. bq. Because if the OverSeer changes and no state event happens there is no way for the node to know it. There is no reason the Overseer needs to pull every 5 seconds to avoid this. We know when a new Overseer is elected. If we keep the 5 second poll of the queue, this issue gains very little IMO.
16. If for some odd reason the Overseer stops being the Overseer without losing it's connection to ZooKeeper, this patch will stop the current Overseer from running when it's alerted the Overseer has changed.
17. Keep in mind, that latch watcher will fire on zk connection events as well as when items are added to the queue.
18. Yeah the Latch Watcher gets a callback , but it the peek(true) would not return unless there is some entry is added to the queue. What happens when the OverSeer changed and no entry is added to the queue ?
19. Ah, right - good point. We don't catch the watcher connection event and have peek fail. The latest change should handle this case though. Either the connection to ZK is gone and the peek call will fail when it retries after getting the connection watcher event. Or the connection might have failed and come back before the retry tries to talk to zk - in this case, the code change I just made should kill the current Overseer threads - either when we are alerted the Overseer has changed or when we start a new overseer because the zk connection was expired.
20. New patch - I was a little off there. For the case where the queue loop does not fail on a down zookeeper because it bounced quickly or something, I said that when we start a new overseer, that will close the old threads - but that only happens if we are elected the overseer. This patch stops any local overseer threads on joining the overseer election.
21. does this patch take care of this scenario? The Overseer went into a GC pause or something and others nodes assume it is down and re-elected another OverSeer. And there is no event in the queue and this OverSeer is still waiting on the queue I don't see the exit condition from the loop in the peek() method

22. bq. The Overseer went into a GC pause or something and others nodes assume it is down and re-elected another OverSeer. This will only happen if our ephemeral overseer leader node goes down - which means we lost our connection to zookeeper. If we lose our connection to zookeeper, the queue thread will exit trying to talk to zk. If we reconnect to zookeeper before the queue loop has a chance to fail (some kind of wicked quick flap), we will stop the overseer threads on getting in line to be the Overseer again. Also, if zk tells us the overseer leader went down, we stop any overseer threads we might have. I think its about as strong as the poll.
23. bq. I don't see the exit condition from the loop in the peek() method You shouldn't need one.
24. You could of course explicitly add one - catch the connection change notification from the watcher and then cause the loop to fail on the next run. I don't think it's strictly necessary, but it wouldn't hurt.
25. Updated patch to trunk changes.
26. **body:** I think that exception handling is better in peek() method better than expecting the consumer of the API to take care of that either way it works
    label: code-design
27. **body:** bq. better than expecting the consumer of the API to take care of that It's really an Overseer specific thing though - I don't think it's a problem for the consumer of the DistributedQueue API. We are dealing with special Overseer considerations, and I don't really see the strategy for that as a consumer. A standard consumer of the queue API does not necessary need to detect this kind of zk connection flap - It's kind of Overseer specific I think. Adding may actually hurt the distributedqueue consumer experience - you expect to keep working the queue in the face of zk connection/disconnection without any special handling.
    label: code-design
28. This patch is even better. It ensures we stop any Overseer threads *before* we reconnect to ZooKeeper on an expiration. This should be even stronger than the 5 second poll.
29. One more patch attached - fixes silly NPE bug when BeforeReconnect was null.
30. Is there anything more we need to do before committing this?
31. Commit 1544255 from [~markrmiller@gmail.com] in branch 'dev/trunk' [https://svn.apache.org/r1544255] SOLR-5436: Eliminate the 1500ms wait in overseer loop as well as polling the ZK distributed queue.
32. Commit 1544257 from [~markrmiller@gmail.com] in branch 'dev/branches/branch_4x' [https://svn.apache.org/r1544257] SOLR-5436: Eliminate the 1500ms wait in overseer loop as well as polling the ZK distributed queue.
33. Why is it not yet resolved?