Item 323
git_comments:

1. update highest sequence persisted
2. Kill and restart broker
3. Set infinite timeout
4. No other messages should be received
5. Repeat the messages and verify they're not received by consumer
6. should trigger complete the batch message, new message will add to a new batch and new batch sequence id use the new message, so that broker can handle the message duplication
7. optional uint64 highest_sequence_id = 24 [default = 0];
8. optional uint64 highest_sequence_id = 6 [default = 0];

git_commits:

1. **summary:** [Issue 5476]Fix message deduplicate issue while using external sequence id with batch produce (#5491)
   message: [Issue 5476]Fix message deduplicate issue while using external sequence id with batch produce (#5491) Fixes #5476 ### Motivation Fix #5476 ### Modifications 1. Add `last_sequence_id` in MessageMetadata and CommandSend, use sequence id and last_sequence_id to indicate the batch `lowest_sequence_id` and `highest_sequence_id`. 2. Handle batch message deduplicate check in MessageDeduplication 3. Response the `last_sequence_id` to client and add message deduplicate check in client

github_issues:

1. **title:** Message deduplication is not well handled when batching is enabled with external provided sequenceId
   body: **Describe the bug** Current implementation of Pulsar producer doesn't check the sequenceId when adding messages to a batch container. That results in violations to idempotent producing with external sequenceId. **To Reproduce** - provide 10 message with sequenceId from 0-9 - provide 10 message with sequenceId from 0-9 again - flush the producer - these 20 messages will be received by the consumer **Expected behavior** The second 10 messages will not be added to container, because they are duplicated. We can throw exceptions to client to indicate that it adds out-of-order sequence ids. **Additional context** There are a couple places requires attentions regarding handling batched messages with external sequenceId. 1) The logic to maintain `lastPublishedSequenceId` is incorrect when using external sequenceId : `lastSequenceIdPublished = op.sequenceId + op.numMessagesInBatch - 1;`. Because the last sequence id is an external sequence id, which can't be computed by adding the number of messages in the batch. 2) We only maintain `lastPublishedSequenceId` (which is the acked seequence id). We also need to maintain a `lastPushSequenceId` to indicate the last sequence id that a producer sends to the broker. 3) the broker need to handle the first sequence id and last sequence id in a message batch.
2. **title:** Message deduplication is not well handled when batching is enabled with external provided sequenceId
   body: **Describe the bug** Current implementation of Pulsar producer doesn't check the sequenceId when adding messages to a batch container. That results in violations to idempotent producing with external sequenceId. **To Reproduce** - provide 10 message with sequenceId from 0-9 - provide 10 message with sequenceId from 0-9 again - flush the producer - these 20 messages will be received by the consumer **Expected behavior** The second 10 messages will not be added to container, because they are duplicated. We can throw exceptions to client to indicate that it adds out-of-order sequence ids. **Additional context** There are a couple places requires attentions regarding handling batched messages with external sequenceId. 1) The logic to maintain `lastPublishedSequenceId` is incorrect when using external sequenceId : `lastSequenceIdPublished = op.sequenceId + op.numMessagesInBatch - 1;`. Because the last sequence id is an external sequence id, which can't be computed by adding the number of messages in the batch. 2) We only maintain `lastPublishedSequenceId` (which is the acked seequence id). We also need to maintain a `lastPushSequenceId` to indicate the last sequence id that a producer sends to the broker. 3) the broker need to handle the first sequence id and last sequence id in a message batch.

github_issues_comments:

github_pulls:

1. **title:** [Issue 5476]Fix message deduplicate issue while using external sequence id with batch produce
   body: Fixes #5476 ### Motivation Fix #5476 ### Modifications 1. Add `last_sequence_id` in MessageMetadata and CommandSend, use sequence id and last_sequence_id to indicate the batch `lowest_sequence_id` and `highest_sequence_id`. 2. Handle batch message deduplicate check in MessageDeduplication 3. Response the `last_sequence_id` to client and add message deduplicate check in client ### Verifying this change Added new unit tests to verify this change ### Does this pull request potentially affect one of the following parts: *If `yes` was chosen, please highlight the changes* - Dependencies (does it add or upgrade a dependency): (no) - The public API: (no) - The schema: (no) - The default values of configurations: (no) - The wire protocol: (yes) - The rest endpoints: (no) - The admin cli options: (no) - Anything that affects deployment: (no) ### Documentation - Does this pull request introduce a new feature? (no)

github_pulls_comments:

1. @sijie It's hard to handle sequence id in key based batcher, if user set an external sequence id, we can't rewrite it and the sequence ids will distributed into inner batchers of key based batcher. Current implementation will throw exception if client can ensure the message is duplicated, since we do not throws exceptions before, is it ok to throws exception?
2. run java8 tests
3. hello @codelipenghui can you fix the test case: ``` org.apache.pulsar.broker.service.ReplicatorTest.testResumptionAfterBacklogRelaxed ```
4. run java8 tests
5. @merlimat Thanks for the review, i have addressed your comments, please take a look again.
6. > org.apache.pulsar.broker.service.PeerReplicatorTest.testPeerClusterInReplicationClusterListChang run java8 tests run integration tests
7. run java8 tests run integration tests
8. @merlimat Please help take a look, i have addressed your comments
9. run java8 tests
10. run java8 tests run cpp tests
11. run java8 tests
12. run java8 tests
13. ping @sijie @merlimat PTAL again. run java8 tests
14. run java8 tests
15. run java8 tests
16. run java8 tests
17. run java8 tests
18. run java8 tests
19. run java8 tests
20. @merlimat can you review this so that we can unblock 2.4.2 release?
21. run java8 tests
22. run java8 tests
23. run java8 tests
24. run cpp tests
25. run java8 tests
26. ping @merlimat PTAL again.
27. run java8 tests
28. run java8 tests
29. run java8 tests
30. ping @merlimat PTAL again.
31. @merlimat PTAL again, thanks
32. @codelipenghui The change is not compatible with 2.4.2, i will move the `Milestone` to `2.5.0`.

github_pulls_reviews:

1. **body:** I don't think this is a valid assumption. The logic of dedup is to transparently perform the deduplication, without giving error to the application in case of duplicates.
   label: code-design

2. **body:** We shouldn't trigger an error to the application when there are dups, rather the contract is that it gets an OK. In any case we need to think through the implication of triggering an error just for one message out of band. This is a very different behavior from the other failure modes where all the messages are failed after 1 failure. I believe that handling this in broker side is much preferable.
   label: code-design
3. If we don't know whether is dup or not, then why are we triggering error here?
4. Why do we need 2 new fields here? Don't we just need to a new `last_sequence_id`? isn't `lowest_sequence_id` the same as `sequence_id` ?
5. I don't see how these sequence ids relate to a `Topic`. * What's the lowest sequenceId ? I think this only apply to 1 single batch. * Sequence id are *per producer* anyway
6. What does it mean the lowest sequence id of a producer?
7. It is in the inner class named MessagePublishContext of Producer
8. If users use the external sequence id and enable batch on producer, sequence id 1,2,3,1 will happens if we do not the check, we can just throw an exception when using the external sequence id?
9. Yes, we can only add a new last_sequence_id, will fix.
10. It is in the inner class named PublishContext of Topic
11. Ok, though as in other comment, there's already a `sequenceId` field. Is that now ignored? If yes, then it should be removed. Also, as you can see, there's an `originalSequenceId` field. This is used in the context of geo-replication and it would have to be accounted for as well.
12. > It is in the inner class named PublishContext of Topic Which already contains a sequence id field
13. This is not correct on 2 levels: 1. If `sequenceId <= lastSequenceIdPushed` we don't know yet whether the message is already dup, because the previous attempt might still fail. This has to be disambiguated by the broker which has visibility at the storage level. 2. As mentioned before, we cannot throw error when there's a duplicate, rather we need to return "ok" to the application.
14. **body:** can this be simplified with the following statement? ``` long callbackSequenceId = Math.max(lastSequenceId, sequenceId); ```
    label: code-design
15. same comment as above
16. **body:** nit: ```suggestion if (lowestSequenceId == -1L) { ```
    label: code-design
17. ```suggestion private long lowestSequenceId = -1L; ```
18. ```suggestion private long highestSequenceId = -1L; ```
19. ```suggestion lowestSequenceId = -1L; ```
20. ```suggestion highestSequenceId = -1L; ```
21. Message with sequence id {} might be a duplicate but cannot be determined at this time.
22. **body:** I would suggest using 'INFO' logging here.
    label: code-design
23. I like the approach here.
24. can you make a common function for this statement here?
25. should this be highest sequence id?
26. 1. I think we need to keep track highestSequenceId, no? 2. We should only assign the highest sequence id, no? ``` lastSequenceIdPushed = Math.max(lastSequenceIdPushed, op.lastSequenceIdPushed); ```
27. It seems to me that we use `highest` sequence id at the client side, but use `last` at the wire protocol and broker side. Can we make it consistent by just using `highest` across the places?
28. **body:** `sequenceId` => `lowestSequenceId` `lastSequenceId` => `highestSequenceId` It is very confusing using `sequenceId` and `lastSequenceId`.
    label: code-design
29. I will add a common function to get the callback sequenceId
30. will fix
31. will fix
32. will fix
33. will fix
34. will fix
35. will fix
36. Ok
37. this should : ``` lastSequenceIdPublished = Math.max(lastSequenceIdPublished, getHighestSequenceId(op)); ``` because the duplicated publishes will still succeed, right. It can override a larger sequence Id with a smaller one.
38. ``` expectedSequenceId = getHighestSequenceId(op); ```

jira_issues:

jira_issues_comments: