[public]
Author: Benjamin Wang
Last update: 13 June 2024

Expected compaction behavior

Background

Watch may drop events quietly when the compact revision happens to be a tombstone revision. Refer to https://github.com/etcd-io/etcd/issues/18089#issuecomment-2158891301 on the root cause and reproduction steps.

We need to clearly clarify the expected compaction behavior.

Three options

When compacting revision X, we can keep or not keep or partially keep the revision X; there are three options.

Option 1: conditionally keep the compact revision

When compacting revision X, we compact all history up to X, and

- keep X if the revision X isn't a tombstone revision.
- Otherwise if revision X is a tombstone revision, compact it as well.

This is the existing implementation.

The good side is it completely keeps the existing behavior, and always cleans up the tombstone revisions within the range during compaction operations.

The problem is that it's hard for the watch server to tell whether a revision is compacted (or even partially compacted) when watch startRevision happens to be the latest compactRevision. So we need to add additional logic in the watch server to handle such a case. But it's hard to add such additional logic if a TXN contains mix operations (i.e put, delete), because part of the data modified in the TXN may be compacted.

Option 2: always keep the compact revision

When compacting revision X,

- compact all history up to X (exclusive), but we always keep the revision X.

- For all revisions which are < X, always keep the latest non-tombstone revision for each key.

It means that users can start to read Revision X and watch starting from X. Reading or Watching X-1 will get an ErrCompacted error.

But it has an impact on HashByRev, please see more details in section “Consistent compaction and Hash” below.

Test cases

unit test on keyIndex
case 1: compact a tombstone revision X, it shouldn't be compacted. Next compaction should cleanup the revision X for both cases below,
1.1 The revision X is the last revision for the key
1.2 The revision X isn't the last revision for the key
case 2: compact a normal revision X, it shouldn't be compacted. In next compaction,
2.1 If the revision X is the last revision for the key, it shouldn't be compacted.
2.2. If the revision X isn't the last revision for the key, it should be compacted.

e2e test on watch
case 1: compact tomestone revisions shouldn't impact watch
rough steps:
1) start a watch client to watch on a key
2) start a goroutine to continuous put or delete a key
3) perform compaction against the different tombstone revisions multiple times, verify that no revisions are dropped.

case 2: compact normal revisions shouldn't impact watch
similar steps as above, but we only execute put operations

case 3: a new watch starting on a compacted tombstone revision should receive the delete event
Refer to the reproduction steps in the issue

e2e test on hash values
case 1: All members should have the same hash value up to the compacted revision after compacting a tombstone revision
rough steps:
1) start a 3 member cluster;
2) execute some put and delete operations;

3) execute compaction on revision X;
4) get hashKV up to revision X from all members, verify the hash values are consistent

1.1 all members have the same etcd version
1.2 mix versions, i.e. main vs 3.5.14 or release-3.5 vs 3.5.14

case 2: all member should have the same hash value up to the compacted revision after compacting a non-tombstone revision
similar steps as above, but execute compact on a non-tombstone revision

2.1 all members have the same etcd version
2.2 mix versions, i.e. main vs 3.5.14 or release-3.5 vs 3.5.14

case 3: All members should have the same hash value up to a non-compacted revision after compacting a tombstone revision
similar steps as case 1, but get hashKv up to a non-compacted revision

3.1 all members have the same etcd version
*3.2 mix versions, i.e. main vs 3.5.14 or release-3.5 vs 3.5.14

case 4: All members should have the same hash value up to a non-compacted revision after compacting a non-tombstone revision
similar steps as case 2, but get hashKv up to a non-compacted revision

4.1 all members have the same etcd version
4.2 mix versions, i.e. main vs 3.5.14 or release-3.5 vs 3.5.14

robustness test
Support compaction, and verify the impact on hash values and watch.

Option 3: doesn't keep the compact revision

When compact a revision X,

- Compact all history up to X (inclusive), and compact the revision X as well if it isn't the latest revision for a key.
- Always keep the latest revision of each key, no matter if it's a tombstone revision or not.

The good side is that it's easy to understand and probably also easy to implement.

The problem is that it breaks the existing behavior. Basically we won't follow this option.

Impact on watch

Note that compaction may impact watch clients, but etcd should never drop any events quietly. The expected behavior for watch clients:

- Either the watch client receives all events: no any events are dropped.
- Or the watch client gets an ErrCompacted response.

Consistent compaction & hash

This is based on discussion between @ahrtr and @fuweid on July 11, 2024.

HashByRev (rev)

- All revisions up to $(rev) should be included. For example, HashByRev(9), all revisions, which <=9, should be included.
- All members should always return the same hash, no matter if it's a mix-version cluster or not.

The values to be returned:

- Previous compact revision
- Hash Rev
- Hash value

Concern

- All members should always return the same hash, no matter if it's a mix-version cluster or not.
- **Solution**: use cluster-level feature-gate to resolve this concern. The fix/change will not be applied until all members have enabled the feature.

Compaction

When compacting revision X,

- compact all history up to X (exclusive), but we always keep the revision X.
- Always keep the latest non-tombstone revision for each key.

It means that users can start to read Revision X and watch starting from X. Reading or Watching X-1 will get an ErrCompacted error.

For example, compact (9),

- Revision 9 will be kept;
- For all revisions, which < 9

- If it’s the latest non-tombstone revision for a key, then it will be kept;
- Otherwise, compact the revisions

Next step

- https://github.com/etcd-io/etcd/pull/18274#discussion_r1672070530
- Ensure we have consensus on the expected behavior.
- Make change per the agreed expected behavior, but it should can be enabled or disabled by a flag. Once the cluster-level feature is ready, migrate the flag to the cluster-level feature.