Item 181
git_comments:

1. 8, Multiple regions, one key range is in a middle region.
2. 5, single region, multiple key ranges
3. * * Generate KeyRange with the given lower Bound (always inclusive) and upper Bound (always exclusive) * @param lowerBound * @param upperBound * @return
4. 11, Multiple regions, multiple key ranges cover all regions.
5. 1, region boundaries is null, return null
6. * 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.
7. 6, Multiple regions, one key range covers all regions.
8. 12, Multiple regions, multiple key ranges cover part regions with cavities and some key ranges are in the same region. Testing boundaries.
9. 4, single regions single key range
10. 10, Multiple regions, one key range (UNBOUND, UNBOUND) covers all regions.
11. 3, empty key ranges, return null
12. 7, Multiple regions, one key range is in the first region.
13. 2, key ranges is null, return null
14. 9, Multiple regions, one key range is in the last region. Testing Boundaries.
15. * * Compare the upper ranges of the two given key ranges * * @param rowKeyRange1 * @param rowKeyRange2 * @return * < 0, the first key range's upper range < the second key range's upper range * = 0, the first key range's upper range = the second key range's upper range * > 0, the first key range's upper range > the second key range's upper range
16. * * Compare the upper ranges of this key range and the given key range. The latter is represented * by (binary array, offset, length, whether it's inclusive or not) * * @param b binary array * @param o offset * @param l length * @param isInclusive is upper range represented in (b, o, l) inclusive or not * @return * < 0, the first key range's upper range < the second key range's upper range * = 0, the first key range's upper range = the second key range's upper range * > 0, the first key range's upper range > the second key range's upper range
17. Move to next key range
18. Move to next boundary
19. * * Split the key ranges into multiple groups along the given boundaries * * @param boundaries * The boundaries is a byte[] list like "b0, b1, ..., bn" which forms * space (UNBOUND, b0), [b0, b1), ..., [bn, UNBOUND). Every boundary * can't be null or empty. * @param keyRanges * The key ranges to split along the given boundaries. Coalesced. * @return * List<Pair<RangeIndex, Query Key Range List in the range>>. * N boundaries split the key space into N+1 ranges. Here each pair * is the index of of such a range and the query key ranges belongs * to that range.

git_commits:

1. **summary:** Merge pull request #485 from BinShi-SecularBird/phoenix-stats
   message: Merge pull request #485 from BinShi-SecularBird/phoenix-stats Add common utility function ScanUtil.splityKeyRangesByBoundaries()

github_issues:

github_issues_comments:

github_pulls:

1. **title:** Add common utility function ScanUtil.splityKeyRangesByBoundaries()

body: #### What does this PR do? Which problem(s) does it fix and how? What does this PR add? #### Where should the reviewer start? Explain what should be set-up first in order to get the review started. Which page has to be viewed etc. #### How should this be manually tested? - [ ] Write down all - [ ] the steps a person - [ ] needs to take to test it. #### Documents ##### Screenshots (if appropriate) If this PR changes something that concerns UI changes, please post before and after screenshots here. ##### Other documents If you have any documents that have anything to do with this PR. #### Who should be notified? - [ ] Write down the name/department this branch needs - [ ] to be communicated to - [ ] Check the box(es) to indicate that it has been - [ ] communicated (add the label) #### What should happen on deployment? (check all that apply) - [x] The usual steps. - [ ] There are database changes, which should be run. - [ ] There are files that should be manually uploaded. #### Questions: - Does this require a blog post? - Does this require a knowledge base update? - Does support need training for this? - Does this has to be communicated to partners?

github_pulls_comments:

1. @twdsilva @dbwong @yanxinyi
2. > Looks good, sorry I wasn't clear what I meant when i said boundaries couldn't be size 0. What I meant was that the elements of boundaries, the individual byte[]'s those cannot be length 0. And maybe add a comment to that. OK, I'll a comment.
3. +1, can you please squash your changes and I will get this committed.
4. @twdsilva, just squashed. Could you check what I did is correct?

github_pulls_reviews:

1. Please add unit test for this function.
2. Any maybe comments on what this is suppose to do for the parameters.
3. This doesn't seem to be configurable consider if static final field makes more sense. Either is okay but as this will be allocated every query maybe keeping it around makes more sense?
4. I dislike the invariant of having an empty list prior to entering the while loop top. What is wrong with just making the new list and then adding it if there are more than 0 entries?
5. Looking in more detail I see that you did this due to how you use continues as flow control which I also dislike and on the continued paths you don't want to increase the index. This makes reasoning/reading the loop more difficult. Looks mostly correct though.
6. May want to comment/test that boundaries cannot be of length 0 due to hbase restriction on keys. This makes comparisons with UNBOUND safe.
7. nit: typo region (!plural)
8. Please consider using a helper function as looking at this all of them use lower inclusive upper exclusive and these will be much more readable.
9. Shouldn't we have trailing empty arrays? In test case 8 we have one leading array but shouldn't we also check the trailing ones exist?
10. I didn't see anything in the transform function ARRAY_TO_LIST that i think is handling this?
11. boundaries' length can be 0.
12. Regarding the first comment, the current design is that, before the last region which has non-empty query key range list, for every region which has no query key range, we generate empty query key range list for it. This is for keeping region index information because it will be used by the caller of this function. I'm considering to return List<Pair<Integer, List<KeyRange>>> represents List<Pair<RegionIndex, Query Key Range List in this region>> which only keeps non-empty query key range list for a region. Regarding the second comment, could you show me more compact and cleaner code without using continue? I'm very curious how it would look like
13. I changed to return List<Pair<Integer, List>>, your first concern should have gone.
14. I'll add comment. BTW, that's the existing function. I just added helper function around the existing function,
15. Added helper function
16. fixed typo
17. I changed to return List<Pair<Integer, List>>, with latest change, this concern should have gone.
18. Added comment. I won't add unit test because this is existing function.
19. Here we have already get a static object by calling ScanUtil.getcomparator(). The modifier static isn't allowed here.
20. Changed to use ASC_FIXED_WIDTH_COMPARATOR directly.

21. should this be List<Pair<BoundaryIndex, Query Key Range List in this boundary>> ?
22. @twdsilva , N boundaries split the key space into N+1 ranges. Here each pair is the index of of such a range and the query key ranges belongs to that range. Probably still use <RangeIndex, Query Key Range List in the range>? Let me add more comment and you can decide how it will be more accurate.

jira_issues:

jira_issues_comments: