Item 24
git_comments:

1. We do not need to do a column reset since we are carefully changing the output.
2. * * Base class for vectorized implementation of GenericUDFMurmurHash. See subclasses for int/string * parameter variations.
3. * * The actual hash method for an item in the input vectors. This implementation is supposed to be * in line with non-vectorized one (ObjectInspectorUtils.getBucketHashCode) for multiple fields. * The method is abstract and is supposed to be implemented in subclasses.
4. if any of the first element is not null, calculate hash
5. otherwise, hash is 0
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. return immediately if batch is empty
8. output of hash will never be null
9. * 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.
10. * * Vectorized expression for hash(Column[int], Column[int]).
11. hash of value from 1. column
12. hash of value from 2. column
13. * 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.
14. * * Vectorized expression for hash(Column[string], Column[int]).
15. hash of value from 1. column
16. hash of value from 2. column
17. * 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.
18. * * Vectorized expression for hash(Column[string], Column[string]).
19. hash of value from 1. column
20. hash of value from 2. column
21. both of the inputs were repeating
22. output's first element is 0, which is hash of null elements
23. output's first element is the hash of first input elements
24. if isRepeating, vectorization logic is not supposed to touch other elements than 0th

25. * 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.
26. fake output value to test short-circuiting
27. non-repeating

git_commits:

1. **summary:** HIVE-23976: Enable vectorization for multi-col semi join reducers (Laszlo Bodor, reviewed by Stamatis Zampetakis, Jesus Camacho Rodriguez)
   message: HIVE-23976: Enable vectorization for multi-col semi join reducers (Laszlo Bodor, reviewed by Stamatis Zampetakis, Jesus Camacho Rodriguez) Closes apache/hive#1458

github_issues:

github_issues_comments:

github_pulls:

1. **title:** HIVE-23976: Enable vectorization for multi-col semi join reducers
   body: ### What changes were proposed in this pull request? Implement vectorized expression for murmur hash. ### Why are the changes needed? Implement vectorized expression for murmur hash in order to stay vectorized for multi-key bloom filter path. ### Does this PR introduce _any_ user-facing change? No, but this is an improvement change, which appears in the explain plan. ### How was this patch tested? Unit test attached. Explain plan shows expected results.

github_pulls_comments:

1. @t3rmin4t0r , @zabetak , @ramesh0201: could you please take a look? good-old vectorization coverage patch, murmurhash with 2 column params
2. This pull request has been automatically marked as stale because it has not had recent activity. It will be closed if no further activity occurs. Feel free to reach out on the dev@hive.apache.org list if the patch is in need of reviews.
3. @abstractdog , @zabetak , this PR has been labeled as stale and is about to be closed automatically. What is the status: Does it need more work or is it ready to be pushed?
4. There are some comments in the JIRA case but we could possibly treat them as follow-ups if necessary. This PR is in good shape already so I think it can be merged as is.
5. @abstractdog , can you trigger tests again?
6. > @abstractdog , can you trigger tests again? yeah, rebased to master, and pushed for new tests

github_pulls_reviews:

jira_issues:

1. **summary:** Enable vectorization for multi-col semi join reducers
   description: HIVE-21196 introduces multi-column semi-join reducers in the query engine. However, the implementation relies on GenericUDFMurmurHash which is not vectorized thus the respective operators cannot be executed in vectorized mode.

jira_issues_comments:

1. Hey [~abstractdog], thanks for taking over this :) I had a quick look on the PR and noticed that the vectorized hash implementation seems to be a binary operator (two inputs, one output) while the non-vectorized alternative (GenericUDFMurmurHash) is an n-ary operator. Can we make the vectorized

implementation n-ary or we should rather transform an expression {{hash(a,b,c,d)}} to something like {{hash(hash(hash(a,b),c),d)}}? In any case if we don't treat this now we should create a follow-up JIRA.

2. [~zabetak]: thanks for taking a look. I think we should go the latter way, as we don't have n-ary vectorized expressions. It used to be a performance decision + now the descriptors don't let us match expression on an "arbitrary" number of input arguments.
3. Hi [~abstractdog], While working on HIVE-24221, I got some further questions/ideas regarding this issue. It seems that we make use of n-ary vectorized expressions for the evaluation of AND and OR operators; its true it is not done with the descriptor but through {{VectorizationContext}}. I am not sure what this mean in terms of efficiency, but it looks like we are saving at least some memory since I get the impression that we can reuse the output vector and not have a different output vector per pair of binary operations. We could employ something similar for an n-ary hash function. Assuming that we cannot/should not treat the hash as n-ary operator then I think it makes more sense to make it unary (single input, single output), instead of binary, being only a kind of wrapper around Murmur for the different datatypes. By doing this the implementation will be simpler and we can cover more use-cases as the combine step is delegated to another abstraction. +Currently+ {noformat} hash(a,b) = 31*murmur(a) + murmur(b) {noformat} +After+ {noformat} hash(a) = murmur(a) {noformat} What do you think?
4. Pushed to master, thanks [~abstractdog]!
5. thanks [~jcamachorodriguez]! as agreed, we will address further improvements in follow-up tickets with [~zabetak]