Item 194
git_comments:

1. * * Tells if the partial/completed match starting at given id should be prunned by given pruningId. * * @param startEventID starting event id of a partial/completed match * @param pruningId pruningId calculated by this strategy * @return true if the match should be pruned
2. * * Retrieves event id of the pruning element from the given match based on the strategy. * * @param match match corresponding to which should the pruning happen * @return pruning event id
3. * * Name of pattern that processing will be skipped to.
4. * * Indicate the skip strategy after a match process.
5. * 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.
6. * * Discards every partial match that contains event of the match preceding the first of *PatternName*. * * @param patternName the pattern name to skip to * @return the created AfterMatchSkipStrategy
7. * * Discards every partial match that contains event of the match. * * @return the created AfterMatchSkipStrategy
8. * * Tells if the strategy may skip some matches. * * @return false if the strategy is NO_SKIP strategy
9. * Forbid further extending.
10. * * Every possible match will be emitted. * * @return the created AfterMatchSkipStrategy
11. * * Discards every partial match that contains event of the match preceding the last of *PatternName*. * * @param patternName the pattern name to skip to * @return the created AfterMatchSkipStrategy
12. * * Prunes matches/partial matches based on the chosen strategy. * * @param matchesToPrune current partial matches * @param matchedResult already completed matches * @param sharedBuffer corresponding shared buffer * @throws Exception thrown if could not access the state
13. * * Every possible match will be emitted.
14. * 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.
15. * * Discards every partial match that contains event of the match.
16. * 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.
17. * * Discards every partial match that contains event of the match preceding the first of *PatternName*.
18. * 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.
19. * * Discards every partial match that contains event of the match preceding the last of *PatternName*.

20. * 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.
21. copy partial matches copy completed matches
22. * Run number (first block in DeweyNumber) -> EventId.
23. * Example from docs.

git_commits:

1. **summary:** [FLINK-9593][cep] Unified After Match semantics with SQL MATCH_RECOGNIZE
   message: [FLINK-9593][cep] Unified After Match semantics with SQL MATCH_RECOGNIZE This closes #6171

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [FLINK-9593] Unified After Match semantics with SQL MATCH_RECOGNIZE
   body: ## What is the purpose of the change Unify semantics of AfteMatch skip with SQL standard to enable CEP ans SQL integration. ## Brief change log - partial/completed matches are pruned based on which one happened first. ## Verifying this change *(Please pick either of the following options)* This change added tests: - testSkipPastLastWithOneOrMoreAtBeginning - testSkipBeforeOtherAlreadyCompleted and adjusted all other tests in class `AfterMatchSkipITCase.java` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (**yes** / no / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know)
2. **title:** [FLINK-9593] Unified After Match semantics with SQL MATCH_RECOGNIZE
   body: ## What is the purpose of the change Unify semantics of AfteMatch skip with SQL standard to enable CEP ans SQL integration. ## Brief change log - partial/completed matches are pruned based on which one happened first. ## Verifying this change *(Please pick either of the following options)* This change added tests: - testSkipPastLastWithOneOrMoreAtBeginning - testSkipBeforeOtherAlreadyCompleted and adjusted all other tests in class `AfterMatchSkipITCase.java` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (**yes** / no / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know)
   label: documentation
3. **title:** [FLINK-9593] Unified After Match semantics with SQL MATCH_RECOGNIZE
   body: ## What is the purpose of the change Unify semantics of AfteMatch skip with SQL standard to enable CEP ans SQL integration. ## Brief change log - partial/completed matches are pruned based on which one happened first. ## Verifying this change *(Please pick either of the following options)* This change added tests: - testSkipPastLastWithOneOrMoreAtBeginning - testSkipBeforeOtherAlreadyCompleted and adjusted all other tests in class `AfterMatchSkipITCase.java` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (**yes** / no / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects

deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know)

4. **title:** [FLINK-9593] Unified After Match semantics with SQL MATCH_RECOGNIZE
   body: ## What is the purpose of the change Unify semantics of AfteMatch skip with SQL standard to enable CEP ans SQL integration. ## Brief change log - partial/completed matches are pruned based on which one happened first. ## Verifying this change *(Please pick either of the following options)* This change added tests: - testSkipPastLastWithOneOrMoreAtBeginning - testSkipBeforeOtherAlreadyCompleted and adjusted all other tests in class `AfterMatchSkipITCase.java` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (**yes** / no / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know)
5. **title:** [FLINK-9593] Unified After Match semantics with SQL MATCH_RECOGNIZE
   body: ## What is the purpose of the change Unify semantics of AfteMatch skip with SQL standard to enable CEP ans SQL integration. ## Brief change log - partial/completed matches are pruned based on which one happened first. ## Verifying this change *(Please pick either of the following options)* This change added tests: - testSkipPastLastWithOneOrMoreAtBeginning - testSkipBeforeOtherAlreadyCompleted and adjusted all other tests in class `AfterMatchSkipITCase.java` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (**yes** / no / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know)
6. **title:** [FLINK-9593] Unified After Match semantics with SQL MATCH_RECOGNIZE
   body: ## What is the purpose of the change Unify semantics of AfteMatch skip with SQL standard to enable CEP ans SQL integration. ## Brief change log - partial/completed matches are pruned based on which one happened first. ## Verifying this change *(Please pick either of the following options)* This change added tests: - testSkipPastLastWithOneOrMoreAtBeginning - testSkipBeforeOtherAlreadyCompleted and adjusted all other tests in class `AfterMatchSkipITCase.java` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (**yes** / no / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know)
7. **title:** [FLINK-9593] Unified After Match semantics with SQL MATCH_RECOGNIZE
   body: ## What is the purpose of the change Unify semantics of AfteMatch skip with SQL standard to enable CEP ans SQL integration. ## Brief change log - partial/completed matches are pruned based on which one happened first. ## Verifying this change *(Please pick either of the following options)* This change added tests: - testSkipPastLastWithOneOrMoreAtBeginning - testSkipBeforeOtherAlreadyCompleted and adjusted all other tests in class `AfterMatchSkipITCase.java` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (**yes** / no / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know)
   label: code-design
8. **title:** [FLINK-9593] Unified After Match semantics with SQL MATCH_RECOGNIZE
   body: ## What is the purpose of the change Unify semantics of AfteMatch skip with SQL standard to enable CEP ans SQL integration. ## Brief change log - partial/completed matches are pruned based on which one happened first. ## Verifying this change *(Please pick either of the following options)* This change added tests: - testSkipPastLastWithOneOrMoreAtBeginning - testSkipBeforeOtherAlreadyCompleted and adjusted all other tests in class `AfterMatchSkipITCase.java` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with

`@Public(Evolving)`: (yes / **no**) - The serializers: (**yes** / no / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know)

9. **title:** [FLINK-9593] Unified After Match semantics with SQL MATCH_RECOGNIZE
   body: ## What is the purpose of the change Unify semantics of AfteMatch skip with SQL standard to enable CEP ans SQL integration. ## Brief change log - partial/completed matches are pruned based on which one happened first. ## Verifying this change *(Please pick either of the following options)* This change added tests: - testSkipPastLastWithOneOrMoreAtBeginning - testSkipBeforeOtherAlreadyCompleted and adjusted all other tests in class `AfterMatchSkipITCase.java` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (**yes** / no / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know)
10. **title:** [FLINK-9593] Unified After Match semantics with SQL MATCH_RECOGNIZE
    body: ## What is the purpose of the change Unify semantics of AfteMatch skip with SQL standard to enable CEP ans SQL integration. ## Brief change log - partial/completed matches are pruned based on which one happened first. ## Verifying this change *(Please pick either of the following options)* This change added tests: - testSkipPastLastWithOneOrMoreAtBeginning - testSkipBeforeOtherAlreadyCompleted and adjusted all other tests in class `AfterMatchSkipITCase.java` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (**yes** / no / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know)

github_pulls_comments:

1. Would appreciate if you had a look @kl0u
2. **body:** Hi, @dawidwys can you explain a little about how does the semantics of `AfterMatch` differ from previous implementation, I read the doc and feel a lille confused. thx ;-)
   label: documentation
3. Thanks @kl0u for review. I've addressed points 1 and 3. As the second one touches some critical parts, let's address it in a separate JIRA.

github_pulls_reviews:

1. What about the `completedMatches`?
2. Same as above.
3. **body:** The name `partialMatches` is misleading because we use it also with the `completedMatches`.
   label: code-design
4. Instead of sorting every time, why not keeping the partial matches in a priority queue?
5. Instead of accessing the state for every match, why not passing all the matches to the shared buffer, and try to fetch the common ones only once. If 2 matches A and B share event with id = 2, we fetch from state only once.

jira_issues:

1. **summary:** Unify AfterMatch semantics with SQL MATCH_RECOGNIZE
   description:
2. **summary:** Unify AfterMatch semantics with SQL MATCH_RECOGNIZE
   description:

jira_issues_comments:

1. GitHub user dawidwys opened a pull request: https://github.com/apache/flink/pull/6171 [FLINK-9593] Unified After Match semantics with SQL MATCH_RECOGNIZE ## What is the purpose of the change

Unify semantics of AfteMatch skip with SQL standard to enable CEP ans SQL integration. ## Brief change log - partial/completed matches are pruned based on which one happened first. ## Verifying this change *(Please pick either of the following options)* This change added tests: - testSkipPastLastWithOneOrMoreAtBeginning - testSkipBeforeOtherAlreadyCompleted and adjusted all other tests in class `AfterMatchSkipITCase.java` ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (yes / **no**) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes / **no**) - The serializers: (**yes** / no / don't know) - The runtime per-record code paths (performance sensitive): (yes / **no** / don't know) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes / **no** / don't know) - The S3 file system connector: (yes / **no** / don't know) You can merge this pull request into a Git repository by running: $ git pull https://github.com/dawidwys/flink cep-after-first-match Alternatively you can review and apply these changes as the patch at: https://github.com/apache/flink/pull/6171.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #6171 ---- commit aca1b71de9b342840043983c8e3eabecb5f0afd4 Author: Dawid Wysakowicz <dwysakowicz@...> Date: 2018-06-14T15:10:05Z [FLINK-9593] Unified After Match semantics with SQL MATCH_RECOGNIZE ----

2. Github user dawidwys commented on the issue: https://github.com/apache/flink/pull/6171 Would appreciate if you had a look @kl0u
3. Github user Aitozi commented on the issue: https://github.com/apache/flink/pull/6171 Hi, @dawidwys can you explain a little about how does the semantics of `AfterMatch` differ from previous implementation, I read the doc and feel a lille confused. thx ;-)
4. Github user kl0u commented on a diff in the pull request: https://github.com/apache/flink/pull/6171#discussion_r198472975 --- Diff: flink-libraries/flink-cep/src/main/java/org/apache/flink/cep/nfa/NFAState.java --- @@ -79,18 +98,18 @@ public boolean equals(Object o) { return false; } NFAState nfaState = (NFAState) o; - return Objects.equals(computationStates, nfaState.computationStates); + return Objects.equals(partialMatches, nfaState.partialMatches); } @Override public int hashCode() { - return Objects.hash(computationStates, stateChanged); + return Objects.hash(partialMatches, stateChanged); --- End diff -- Same as above.
5. Github user kl0u commented on a diff in the pull request: https://github.com/apache/flink/pull/6171#discussion_r198473858 --- Diff: flink-libraries/flink-cep/src/main/java/org/apache/flink/cep/nfa/NFA.java --- @@ -330,77 +328,85 @@ private boolean isStateTimedOut(final ComputationState state, final long timesta } } - discardComputationStatesAccordingToStrategy(- sharedBuffer, computationStates, result, afterMatchSkipStrategy); + if (!potentialMatches.isEmpty()) { + nfaState.setStateChanged(); + } + + List<Map<String, List<T>>> result = new ArrayList<>(); + if (afterMatchSkipStrategy.isSkipStrategy()) { + processMatchesAccordingToSkipStrategy(sharedBuffer, + nfaState, + afterMatchSkipStrategy, + potentialMatches, + result); + } else { + for (ComputationState match : potentialMatches) { + result.add(sharedBuffer.materializeMatch(sharedBuffer.extractPatterns(match.getPreviousBufferEntry(), match.getVersion()).get(0))); + sharedBuffer.releaseNode(match.getPreviousBufferEntry()); + } + } return result; } - private void discardComputationStatesAccordingToStrategy(- final SharedBuffer<T> sharedBuffer, - final Queue<ComputationState> computationStates, - final Collection<Map<String, List<T>>> matchedResult, - final AfterMatchSkipStrategy afterMatchSkipStrategy) throws Exception { + private void processMatchesAccordingToSkipStrategy(+ SharedBuffer<T> sharedBuffer, + NFAState nfaState, + AfterMatchSkipStrategy afterMatchSkipStrategy, + PriorityQueue<ComputationState> potentialMatches, + List<Map<String, List<T>>> result) throws Exception { - Set<T> discardEvents = new HashSet<>(); - switch(afterMatchSkipStrategy.getStrategy()) { - case SKIP_TO_LAST: - for (Map<String, List<T>> resultMap: matchedResult) { - for (Map.Entry<String, List<T>> keyMatches : resultMap.entrySet()) { - if (keyMatches.getKey().equals(afterMatchSkipStrategy.getPatternName())) { - discardEvents.addAll(keyMatches.getValue().subList(0, keyMatches.getValue().size() - 1)); - break; - } else { - discardEvents.addAll(keyMatches.getValue()); - } - } - } - break; - case SKIP_TO_FIRST: - for (Map<String, List<T>> resultMap: matchedResult) { - for (Map.Entry<String, List<T>> keyMatches : resultMap.entrySet()) { - if (keyMatches.getKey().equals(afterMatchSkipStrategy.getPatternName())) { - break; - } else { - discardEvents.addAll(keyMatches.getValue()); - } - } - } - break; - case SKIP_PAST_LAST_EVENT: - for (Map<String, List<T>> resultMap: matchedResult) { - for (List<T> eventList: resultMap.values()) { - discardEvents.addAll(eventList); - } - } - break; - } - if (!discardEvents.isEmpty()) { - List<ComputationState> discardStates = new ArrayList<>(); - for (ComputationState computationState : computationStates) { - boolean discard = false; - Map<String, List<T>> partialMatch = extractCurrentMatches(sharedBuffer, computationState); - for (List<T> list:

partialMatch.values()) { - for (T e: list) { - if (discardEvents.contains(e)) { - // discard the computation state. - discard = true; - break; - } - } - if (discard) { - break; - } - } - if (discard) { - sharedBuffer.releaseNode(computationState.getPreviousBufferEntry()); - discardStates.add(computationState); - } + nfaState.getCompletedMatches().addAll(potentialMatches); + + ComputationState earliestMatch = nfaState.getCompletedMatches().peek(); + + if (earliestMatch != null) { + Queue<ComputationState> sortedPartialMatches = sortByStartTime(nfaState.getPartialMatches()); --- End diff -- Instead of sorting every time, why not keeping the partial matches in a priority queue?

6. Github user kl0u commented on a diff in the pull request: https://github.com/apache/flink/pull/6171#discussion_r198474417 --- Diff: flink-libraries/flink-cep/src/main/java/org/apache/flink/cep/nfa/NFA.java --- @@ -330,77 +328,85 @@ private boolean isStateTimedOut(final ComputationState state, final long timesta } } - discardComputationStatesAccordingToStrategy(- sharedBuffer, computationStates, result, afterMatchSkipStrategy); + if (!potentialMatches.isEmpty()) { + nfaState.setStateChanged(); + } + + List<Map<String, List<T>>> result = new ArrayList<>(); + if (afterMatchSkipStrategy.isSkipStrategy()) { + processMatchesAccordingToSkipStrategy(sharedBuffer, + nfaState, + afterMatchSkipStrategy, + potentialMatches, + result); + } else { + for (ComputationState match : potentialMatches) { + result.add(sharedBuffer.materializeMatch(sharedBuffer.extractPatterns(match.getPreviousBufferEntry(), --- End diff -- Instead of accessing the state for every match, why not passing all the matches to the shared buffer, and try to fetch the common ones only once. If 2 matches A and B share event with id = 2, we fetch from state only once.
7. Github user kl0u commented on a diff in the pull request: https://github.com/apache/flink/pull/6171#discussion_r198472927 --- Diff: flink-libraries/flink-cep/src/main/java/org/apache/flink/cep/nfa/NFAState.java --- @@ -79,18 +98,18 @@ public boolean equals(Object o) { return false; } NFAState nfaState = (NFAState) o; - return Objects.equals(computationStates, nfaState.computationStates); + return Objects.equals(partialMatches, nfaState.partialMatches); --- End diff -- What about the `completedMatches`?
8. Github user kl0u commented on a diff in the pull request: https://github.com/apache/flink/pull/6171#discussion_r198473426 --- Diff: flink-libraries/flink-cep/src/main/java/org/apache/flink/cep/nfa/aftermatch/AfterMatchSkipStrategy.java --- @@ -0,0 +1,155 @@ +/* + * 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. + */ + +package org.apache.flink.cep.nfa.aftermatch; + +import org.apache.flink.cep.nfa.ComputationState; +import org.apache.flink.cep.nfa.sharedbuffer.EventId; +import org.apache.flink.cep.nfa.sharedbuffer.SharedBuffer; + +import java.io.Serializable; +import java.util.ArrayList; +import java.util.Collection; +import java.util.List; +import java.util.Map; +import java.util.Optional; + + +/** + * Indicate the skip strategy after a match process. + */ +public abstract class AfterMatchSkipStrategy implements Serializable { + + private static final long serialVersionUID = -4048930333619068531L; + + /** + * Discards every partial match that contains event of the match preceding the first of *PatternName*. + * + * @param patternName the pattern name to skip to + * @return the created AfterMatchSkipStrategy + */ + public static AfterMatchSkipStrategy skipToFirst(String patternName) { + return new SkipToFirstStrategy(patternName); + } + + /** + * Discards every partial match that contains event of the match preceding the last of *PatternName*. + * + * @param patternName the pattern name to skip to + * @return the created AfterMatchSkipStrategy + */ + public static AfterMatchSkipStrategy skipToLast(String patternName) { + return new SkipToLastStrategy(patternName); + } + + /** + * Discards every partial match that contains event of the match. + * + * @return the created AfterMatchSkipStrategy + */ + public static AfterMatchSkipStrategy skipPastLastEvent() { + return SkipPastLastStrategy.INSTANCE; + } + + /** + * Every possible match will be emitted. + * + * @return the created AfterMatchSkipStrategy + */ + public static AfterMatchSkipStrategy noSkip() { + return NoSkipStrategy.INSTANCE; + } + + /** + * Tells if the strategy may skip some matches. + * + * @return false if the strategy is NO_SKIP strategy + */ + public abstract boolean isSkipStrategy(); + + /** + * Prunes matches/partial matches based on the chosen strategy. + * + * @param partialMatches current partial matches + * @param matchedResult already

completed matches + * @param sharedBuffer corresponding shared buffer + * @throws Exception thrown if could not access the state + */ + public void prune(+ Collection<ComputationState> partialMatches, --- End diff -- The name `partialMatches` is misleading because we use it also with the `completedMatches`.

9. Github user dawidwys commented on the issue: https://github.com/apache/flink/pull/6171 Thanks @kl0u for review. I've addressed points 1 and 3. As the second one touches some critical parts, let's address it in a separate JIRA.
10. Github user asfgit closed the pull request at: https://github.com/apache/flink/pull/6171