Item 191
git_comments:

1. * * Formed of a mandatory operator ID and optionally a user defined operator ID.
2. * 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.
3. * The IDs of all operators contained in this vertex. * * <p>The ID pairs are stored depth-first post-order; for the forking chain below the ID's would be stored as [D, E, B, C, A]. * A - B - D * \ \ * C E * This is the same order that operators are stored in the {@code StreamTask}.

git_commits:

1. **summary:** [FLINK-16638][runtime][checkpointing] Flink checkStateMappingCompleteness doesn't include UserDefinedOperatorIDs
   message: [FLINK-16638][runtime][checkpointing] Flink checkStateMappingCompleteness doesn't include UserDefinedOperatorIDs

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Flink checkStateMappingCompleteness doesn't include UserDefinedOperatorIDs
   description: [StateAssignmentOperation.checkStateMappingCompleteness|https://github.com/apache/flink/blob/master/flink-runtime/src/main/java/org/apache/flink/runtime/checkpoint/StateAssignmentOperation.java#L555] doesn't check for UserDefinedOperatorIDs (specified using setUidHash), causing the exception: {code} java.lang.IllegalStateException: There is no operator for the state {} {code} to be thrown when a savepoint can't be mapped to an ExecutionJobVertex, even when the operator hash is explicitly specified. I believe this logic should be extended to also include UserDefinedOperatorIDs as so: {code:java} for (ExecutionJobVertex executionJobVertex : tasks) { allOperatorIDs.addAll(executionJobVertex.getOperatorIDs()); allOperatorIDs.addAll(executionJobVertex.getUserDefinedOperatorIDs()); } {code}

jira_issues_comments:

1. Thanks for reporting this issue [~basharaj], You're right the allOperatorIDs contains non-user-defined IDs. However, I see that the other side of the check is also using non-user-defined IDs (StateAssignmentOperation#operatorStates). So the code looks fine to me. Probably, the snapshot data is corrupted. The log message you provided looks strange: it should contain the operator ID that can't be mapped. In your case it's just {}. Can you please post a full message with a stacktrace?
2. Hi [~roman_khachatryan], sorry I didn't post the full log message, the error is: {code:java} java.lang.IllegalStateException: There is no operator for the state 6463bd1ad519d1e0c283c83f761989c1 at org.apache.flink.runtime.checkpoint.StateAssignmentOperation.checkStateMappingCompleteness(StateAssignmentOperation.java:567) at org.apache.flink.runtime.checkpoint.StateAssignmentOperation.assignStates(StateAssignmentOperation.java:79) at org.apache.flink.runtime.checkpoint.CheckpointCoordinator.restoreLatestCheckpointedState(CheckpointCoordinator.java:1078) at org.apache.flink.runtime.checkpoint.CheckpointCoordinator.restoreSavepoint(CheckpointCoordinator.java:1143) at org.apache.flink.runtime.scheduler.LegacyScheduler.tryRestoreExecutionGraphFromSavepoint(LegacyScheduler.java:237) at org.apache.flink.runtime.scheduler.LegacyScheduler.createAndRestoreExecutionGraph(LegacyScheduler.java:196) at org.apache.flink.runtime.scheduler.LegacyScheduler.<init>(LegacyScheduler.java:176) at org.apache.flink.runtime.scheduler.LegacySchedulerFactory.createInstance(LegacySchedulerFactory.java:70) at org.apache.flink.runtime.jobmaster.JobMaster.createScheduler(JobMaster.java:275) at org.apache.flink.runtime.jobmaster.JobMaster.<init>(JobMaster.java:265) at org.apache.flink.runtime.jobmaster.factories.DefaultJobMasterServiceFactory.createJobMasterService(DefaultJobMasterServiceFactory.java:98) at org.apache.flink.runtime.jobmaster.factories.DefaultJobMasterServiceFactory.createJobMasterService(DefaultJobMasterServiceFactory.java:40) at org.apache.flink.runtime.jobmaster.JobManagerRunner.<init>(JobManagerRunner.java:146) {code} I still don't see how the code is right. Operator hash 6463bd1ad519d1e0c283c83f761989c1 is in the save point so it's in _operatorStates_. When I set that hash directly on the operator using setUidHash the code is not adding it (through a call to getUserDefinedOperatorIDs) to `allOperatorIDs` and this condition will always be true {code:java} if (!allOperatorIDs.contains(operatorGroupStateEntry.getKey())) { {code} Throwing the exception above.
3. Unfortunately, I can't follow the stacktrace (e.g. LegacyScheduler). Are you sure your version is 1.10?
4. [~roman_khachatryan] sorry this was against 1.9.1, I updated the Jira.
5. Thanks, [~basharaj]. I think you are right, this is a bug.
6. [~basharaj], do you want to contribute the PR for this bug? If so, i can assign this ticket for you. :)
7. [~zjwang] sure I can.
8. [~basharaj] thanks for reporting and volunteering :) Just a quick question what's the timeline that you could provide a fix for this?
9. [~pnowojski]I I can submit a fix by end of this week, would that be OK?
10. Sure, thanks!
11. Hey [~basharaj], could we get another estimate when could you work on this ticket? Sorry for putting a pressure, but we would like it to be fixed for 1.11 release. If you can not work on it in the next week or so, someone might need to take it over.
12. [~pnowojski] sorry about that. Unfortunately due to work and personal reasons I couldn't get a chance to work on this. Please feel free to assign it to someone else to meet the 1.11 release schedule.
13. Thanks for the update [~basharaj]. No problem. I have unassigned you for a time being. I'm not sure exactly when someone from our side would be able to pick it up, so if you find some time feel free to come back to this issue.
14. **body:** Hi [~pnowojski] and [~basharaj] I volunteer to fix this one, Bashar already offered most of the "just do it" solution. I also suggest this opportunity to clean up the code. [JobVertex|https://github.com/apache/flink/blob/a9f8a0b4481719fb511436a61e36cb0e26559c79/flink-runtime/src/main/java/org/apache/flink/runtime/jobgraph/JobVertex.java#L59] has a field called _idAlternatives_ that is only used in a test and in a separate function that is also only invoked from a unit test (i.e. it can be deleted). It also has fields _operatorIDs_ and _operatorIdsAlternatives_, their getters are called in similar situations, _operatorIdsAlternatives_'s getter has an extra invocation in [ExecutionJobVertex::includeAlternativeOperatorIDs|https://github.com/apache/flink/blob/a9f8a0b4481719fb511436a61e36cb0e26559c79/flink-runtime/src/main/java/org/apache/flink/runtime/executiongraph/ExecutionJobVertex.java#L667] that is only making up for a previous call to get only the _operatorIDs_. It seems like both concepts are tightly related and should be encapsulated in a single entity. Not surprisingly the bug we're tackling here is due to retrieving one set of ids while forgetting the other set of ids. There are 6 production code invocations to [ExecutionJobVertex::getOperatorIDs|https://github.com/apache/flink/blob/a9f8a0b4481719fb511436a61e36cb0e26559c79/flink-runtime/src/main/java/org/apache/flink/runtime/executiongraph/ExecutionJobVertex.java#L296]: - The call in

[Checkpoints::loadAndValidateCheckpoint|#L140] is soon followed by the previously mentioned _ExecutionJobVertex::includeAlternativeOperatorIDs_. Reinforcing the argument that both are related. * The call in [StateAssignmentOperation::assignStates|https://github.com/apache/flink/blob/a9f8a0b4481719fb511436a61e36cb0e26559c79/flink-runtime/src/main/java/org/apache/flink/runtime/checkpoint/StateAssignmentOperation.java#L89] is directly followed to call to retrieve the operator id alternatives, also reinforcing the argument. * The call in [StateAssignmentOperation::checkStateMappingCompleteness|https://github.com/apache/flink/blob/a9f8a0b4481719fb511436a61e36cb0e26559c79/flink-runtime/src/main/java/org/apache/flink/runtime/checkpoint/StateAssignmentOperation.java#L558] is the one with the bug! - The other 3 in [StateAssignmentOperation::assignTaskStateToExecutionJobVertices|https://github.com/apache/flink/blob/a9f8a0b4481719fb511436a61e36cb0e26559c79/flink-runtime/src/main/java/org/apache/flink/runtime/checkpoint/StateAssignmentOperation.java#L217], [StateAssignmentOperation::assignAttemptState|https://github.com/apache/flink/blob/a9f8a0b4481719fb511436a61e36cb0e26559c79/flink-runtime/src/main/java/org/apache/flink/runtime/checkpoint/StateAssignmentOperation.java#L118], and [PendingCheckpoint::acknowledgeTask|https://github.com/apache/flink/blob/a9f8a0b4481719fb511436a61e36cb0e26559c79/flink-runtime/src/main/java/org/apache/flink/runtime/checkpoint/PendingCheckpoint.java#L374] are not followed by a retrieval of the alternative ids nor do they need one. Why these don't need to is puzzling me and they're the only reason not to encapsulate both ids together. If someone could explain this we might be able to leave the code pretty tidy! I offer myself volunteer for either the quick dirty fix or the longer cleaner solution. The first should take me a couple of days whereas the second should take me ~4 days (considering I've already done a lot of the ground work locally). I'm open to suggestions.
label: code-design

15. Hi [~edu05], Thanks a lot for offering your help! Can you please describe the second approach that you mentioned?
16. Hi [~roman_khachatryan], sure, the idea is to avoid future similar bugs by forcing developers to access a single list of (operatorId, userDefinedOperatorId) pairs. Using only one set of ids or both is then a conscious choice of the developer. This can be achieved with a new class OperatorIdPair that will hold both ids. I've pushed the putative changes to [https://github.com/edu05/flink/tree/FLINK-16638-PREP] , the fix is also included, developers are forced to make a conscious decision as which operator ids to use and presents the chance to remove some code (look at //COULD THIS IF BLOCK BE REMOVED IF THE ALTERNATIVE IDS HAD BEEN ADDED TO THE MAPPING FROM THE START? comment). My question in my previous post was around how come there are 3 out of 6 invocations that DO need the user defined operator id, whereas the other 3 do NOT. I hope this helps illustrate, let me know your thoughts!
17. Thanks Eduardo, I agree OperatorID pair makes sense. Can you please create a PR? We can also fix the issues you mentioned there.
18. [~roman_khachatryan] Sure will! Expect one in one or two days :)
19. Hi [~roman_khachatryan] I've submitted a PR, it's on the last stage of the Azure build (e2e tests). Let me know if I can be of any help with the review.
20. Hi [~edu05], I'll take a look. Thanks!
21. Thanks for the contribution [~edu05] and for reporting [~basharaj]. merged commit 0114338 into apache:master