Item 197
git_comments:

1. * 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.
2. * * Handler for exceptions that happen on checkpointing. The handler can reject and rethrow the offered exceptions.
3. * * Offers the exception for handling. If the exception cannot be handled from this instance, it is rethrown. * * @param checkpointMetaData metadata for the checkpoint for which the exception occurred. * @param exception the exception to handle. * @throws Exception rethrows the exception if it cannot be handled.
4. * * This handler makes the task fail by rethrowing a reported exception.
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. * * This handler makes the task decline the checkpoint as reaction to the reported exception. The task is not failed.
7. * * Returns a {@link CheckpointExceptionHandler} that either causes a task to fail completely or to just declines * checkpoint on exception, depending on the parameter flag.
8. * * This factory produces {@link CheckpointExceptionHandler} instances that handle exceptions during checkpointing in a * {@link StreamTask}.
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. * * Test that the configuration mechanism for how tasks react on checkpoint errors works correctly.
11. * 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.
12. * * Test for the current implementations of {@link CheckpointExceptionHandler} and their factory.
13. start the task and wait until it is in "restore"
14. test source operator that calls into the locking checkpoint output stream.
15. ---------- state backends that trigger errors in checkpointing. ----------
16. ---------- state backend with locking output stream. ----------
17. checkpoint responder that records a call to decline.

18. * * This method is visible because of the way the configuration is currently forwarded from the checkpoint config to * the task. This should not be called by the user, please use CheckpointConfig.isFailTaskOnCheckpointError() * instead.
19. * * This method is visible because of the way the configuration is currently forwarded from the checkpoint config to * the task. This should not be called by the user, please use CheckpointConfig.setFailOnCheckpointingErrors(...) * instead.
20. * Determines if a task fails or not if there is an error in writing its checkpoint data. Default: true
21. * * This determines the behaviour of tasks if there is an error in their local checkpointing. If this returns true, * tasks will fail as a reaction. If this returns false, task will only decline the failed checkpoint.
22. * Determines if a tasks are failed or not if there is an error in their checkpointing. Default: true
23. * * Sets the expected behaviour for tasks in case that they encounter an error in their checkpointing procedure. * If this is set to true, the task will fail on checkpointing error. If this is set to false, the task will only * decline a the checkpoint and continue running. The default is true.
24. propagate the expected behaviour for checkpoint errors to task.
25. * Handler for exceptions during checkpointing in the stream task. Used in synchronous part of the checkpoint.
26. * * Wrapper for synchronous {@link CheckpointExceptionHandler}. This implementation catches unhandled, rethrown * exceptions and reports them through {@link #handleAsyncException(String, Throwable)}. As this implementation * always handles the exception in some way, it never rethrows.
27. **comment:** we are transferring ownership over snapshotInProgressList for cleanup to the thread, active on submit
    label: code-design
28. * Owning stream task to which we report async exceptions.
29. **comment:** * Wrapper for synchronousCheckpointExceptionHandler to deal with rethrown exceptions. Used in the async part.
    label: code-design
30. **comment:** * Synchronous exception handler to which we delegate.
    label: code-design
31. start the task and wait until it is in "restore"
32. test source operator that calls into the locking checkpoint output stream.
33. ---------- state backends that trigger errors in checkpointing. ----------
34. ---------- state backend with locking output stream. ----------
35. checkpoint responder that records a call to decline.

git_commits:

1. **summary:** [FLINK-4809] [checkpoints] Operators should tolerate checkpoint failures.
   message: [FLINK-4809] [checkpoints] Operators should tolerate checkpoint failures. This closes #4883.

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [FLINK-4809] Operators should tolerate checkpoint failures.
   body: ## What is the purpose of the change This PR implements FLINK-4809 and allows tasks to tolerate errors that happen in their checkpointing procedure. This tolerance is optional and can activated through `CheckpointOptions`, the default behaviour is to fail the task, as before this PR. When fault tolerance is activated, an error will not fail the task. Instead, the task will decline the checkpoint to the checkpoint coordinator. ## Brief change log - *Feature configuration through `CheckpointOptions` and `ExecutionConfig`.* - *Introduced `CheckpointExceptionHandler` and integrated with StreamTask. This handler will either transform the exception to a `declineCheckpoint` or rethrow to fail the task. * - *Tests for configuration forwarding and functionality.* ## Verifying this change This change added tests and can be verified as follows: - *`CheckpointExceptionHandlerTest`: Unit test for the implementations of `CheckpointExceptionHandlerTest` and their factory.* - *`TaskCheckpointingBehaviourTest`: Tests the functionality to either fail tasks or only decline checkpoints in case of failure.* -

`CheckpointExceptionHandlerConfigurationTest`: Checks everything about configuration and configuration forwarding for the feature. ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (no) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes) - The serializers: (no) - The runtime per-record code paths (performance sensitive): (no) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes) ## Documentation - Does this pull request introduce a new feature? (yes) - If yes, how is the feature documented? (docs and JavaDocs)

github_pulls_comments:

1. CC @aljoscha
2. @StefanRRichter do we have any update on this PR?
3. @PangZhi No updates, because I was on vacation and we currently give priority to the release 1.4 testing. After that I will finish this, but from the review comments, I would not expect that there are many changes to what I implemented and this can be merged as soon as there is time to do so.
4. @aljoscha Thanks for the review, I addressed your points. Will merge this.

github_pulls_reviews:

1. Duplicate option in `ExecutionConfig` and `CheckpointingConfig`.
2. I think we typically don't talk about tasks in the user facing APIs. This could be `isFailOnCheckpointingErrors()`, for example.
3. **body:** nit: "exception handles"
   label: code-design
4. Why don't we check for `failed` anymore?
5. Because this was moved from the `finally`-block into a `catch`-block where it is clear that the code failed.
6. After discussions with @aljoscha, we decided to keep this because it is the current way that Flink forwards the configuration. We add some `@Internal` annotation to the methods in `ExecutionConfig` so that the user will use the proper calls on `CheckpointingConfig` instead.
7. This line is missing from the Java tab.

jira_issues:

1. **summary:** Operators should tolerate checkpoint failures
   description: Operators should try/catch exceptions in the synchronous and asynchronous part of the checkpoint and send a {{DeclineCheckpoint}} message as a result. The decline message should have the failure cause attached to it. The checkpoint barrier should be sent anyways as a first step before attempting to make a state checkpoint, to make sure that downstream operators do not block in alignment.

jira_issues_comments:

1. GitHub user StefanRRichter opened a pull request: https://github.com/apache/flink/pull/4883 [FLINK-4809] Operators should tolerate checkpoint failures. ## What is the purpose of the change This PR implements FLINK-4809 and allows tasks to tolerate errors that happen in their checkpointing procedure. This tolerance is optional and can activated through `CheckpointOptions`, the default behaviour is to fail the task, as before this PR. When fault tolerance is activated, an error will not fail the task. Instead, the task will decline the checkpoint to the checkpoint coordinator. ## Brief change log - *Feature configuration through `CheckpointOptions` and `ExecutionConfig`.* - *Introduced `CheckpointExceptionHandler` and integrated with StreamTask. This handler will either transform the exception to a `declineCheckpoint` or rethrow to fail the task. * - *Tests for configuration forwarding and functionality.* ## Verifying this change This change added tests and can be verified as follows: - *`CheckpointExceptionHandlerTest`: Unit test for the implementations of `CheckpointExceptionHandlerTest` and their factory.* - *`TaskCheckpointingBehaviourTest`: Tests the functionality to either fail tasks or only decline checkpoints in case of failure.* - *`CheckpointExceptionHandlerConfigurationTest`: Checks everything about configuration and configuration forwarding for the feature.* ## Does this pull request potentially affect one of the following parts: - Dependencies (does it add or upgrade a dependency): (no) - The public API, i.e., is any changed class annotated with `@Public(Evolving)`: (yes) - The serializers: (no) - The runtime per-record code paths (performance sensitive): (no) - Anything that affects deployment or recovery: JobManager (and its components), Checkpointing, Yarn/Mesos, ZooKeeper: (yes) ## Documentation - Does this pull request

introduce a new feature? (yes) - If yes, how is the feature documented? (docs and JavaDocs) You can merge this pull request into a Git repository by running: $ git pull https://github.com/StefanRRichter/flink decline-on-checkpoint-exception Alternatively you can review and apply these changes as the patch at: https://github.com/apache/flink/pull/4883.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 #4883 ---- commit 9156a70319f4227b2a07edd9b07dae5fcfbaa01c Author: Stefan Richter <s.richter@data-artisans.com> Date: 2017-10-20T08:59:45Z [FLINK-4809] Operators should tolerate checkpoint failures. ----

2. Github user StefanRRichter commented on the issue: https://github.com/apache/flink/pull/4883 CC @aljoscha
3. Github user aljoscha commented on a diff in the pull request: https://github.com/apache/flink/pull/4883#discussion_r146544396 --- Diff: flink-streaming-java/src/main/java/org/apache/flink/streaming/runtime/tasks/StreamTask.java --- @@ -1041,27 +1065,27 @@ public void executeCheckpointing() throws Exception { checkpointMetrics.getAlignmentDurationNanos() / 1_000_000, checkpointMetrics.getSyncDurationMillis()); } - } finally { - if (failed) { - // Cleanup to release resources - for (OperatorSnapshotResult operatorSnapshotResult : operatorSnapshotsInProgress.values()) { - if (null != operatorSnapshotResult) { - try { - operatorSnapshotResult.cancel(); - } catch (Exception e) { - LOG.warn("Could not properly cancel an operator snapshot result.", e); - } + } catch (Exception ex) { + // Cleanup to release resources --- End diff -- Why don't we check for `failed` anymore?
4. Github user aljoscha commented on a diff in the pull request: https://github.com/apache/flink/pull/4883#discussion_r146543421 --- Diff: flink-core/src/main/java/org/apache/flink/api/common/ExecutionConfig.java --- @@ -150,6 +150,9 @@ /** This flag defines if we use compression for the state snapshot data or not. Default: false */ private boolean useSnapshotCompression = false; + /** Determines if a task fails or not if there is an error in writing its checkpoint data. Default: true */ --- End diff -- Duplicate option in `ExecutionConfig` and `CheckpointingConfig`.
5. Github user aljoscha commented on a diff in the pull request: https://github.com/apache/flink/pull/4883#discussion_r146543822 --- Diff: flink-streaming-java/src/main/java/org/apache/flink/streaming/runtime/tasks/CheckpointExceptionHandlerFactory.java --- @@ -0,0 +1,78 @@ +/* + * 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.streaming.runtime.tasks; +import org.apache.flink.runtime.checkpoint.CheckpointMetaData; +import org.apache.flink.runtime.execution.Environment; +import org.apache.flink.util.Preconditions; + +/** + * This factory produces exception handles that handle exceptions during checkpointing in a {@link StreamTask}. --- End diff -- nit: "exception handles"
6. Github user aljoscha commented on a diff in the pull request: https://github.com/apache/flink/pull/4883#discussion_r146543640 --- Diff: flink-streaming-java/src/main/java/org/apache/flink/streaming/api/environment/CheckpointConfig.java --- @@ -231,6 +234,23 @@ public void setForceCheckpointing(boolean forceCheckpointing) { } /** + * This determines the behaviour of tasks if there is an error in their checkpointing. If this returns true, + * tasks will fail as a reaction. If this returns false, task will only decline the failed checkpoint. + */ + public boolean isFailTasksOnCheckpointingErrors() { --- End diff -- I think we typically don't talk about tasks in the user facing APIs. This could be `isFailOnCheckpointingErrors()`, for example.
7. Do we have any update on the PR? It has been hanging for weeks.
8. Github user PangZhi commented on the issue: https://github.com/apache/flink/pull/4883 @StefanRRichter do we have any update on this PR?
9. Github user StefanRRichter commented on the issue: https://github.com/apache/flink/pull/4883 @PangZhi No updates, because I was on vacation and we currently give priority to the release 1.4 testing. After that I will finish this, but from the review comments, I would not expect that there are many changes to what I implemented and this can be merged as soon as there is time to do so.
10. Github user StefanRRichter commented on a diff in the pull request: https://github.com/apache/flink/pull/4883#discussion_r151940619 --- Diff: flink-streaming-

java/src/main/java/org/apache/flink/streaming/runtime/tasks/StreamTask.java --- @@ -1041,27 +1065,27 @@ public void executeCheckpointing() throws Exception { checkpointMetrics.getAlignmentDurationNanos() / 1_000_000, checkpointMetrics.getSyncDurationMillis()); } - } finally { - if (failed) { - // Cleanup to release resources - for (OperatorSnapshotResult operatorSnapshotResult : operatorSnapshotsInProgress.values()) { - if (null != operatorSnapshotResult) { - try { - operatorSnapshotResult.cancel(); - } catch (Exception e) { - LOG.warn("Could not properly cancel an operator snapshot result.", e); - } + } catch (Exception ex) { + // Cleanup to release resources --- End diff -- Because this was moved from the `finally`-block into a `catch`-block where it is clear that the code failed.

11. Github user StefanRRichter commented on a diff in the pull request: https://github.com/apache/flink/pull/4883#discussion_r151948416 --- Diff: flink-core/src/main/java/org/apache/flink/api/common/ExecutionConfig.java --- @@ -150,6 +150,9 @@ /** This flag defines if we use compression for the state snapshot data or not. Default: false */ private boolean useSnapshotCompression = false; + /** Determines if a task fails or not if there is an error in writing its checkpoint data. Default: true */ --- End diff -- After discussions with @aljoscha, we decided to keep this because it is the current way that Flink forwards the configuration. We add some `@Internal` and `@Deprecated` annotations to the methods in `ExecutionConfig` so that the user will use the proper calls on `CheckpointingConfig` instead.
12. Github user StefanRRichter commented on the issue: https://github.com/apache/flink/pull/4883 @aljoscha Thanks for the review, I addressed your points. Will merge this.
13. Github user asfgit closed the pull request at: https://github.com/apache/flink/pull/4883
14. Merged in {{[7c63526|https://github.com/apache/flink/commit/7c63526ad6f27c6f15625b8b6c48359d9532890b].}}
15. Merged in {{[7c63526|https://github.com/apache/flink/commit/7c63526ad6f27c6f15625b8b6c48359d9532890b].}}
16. Github user rmetzger commented on a diff in the pull request: https://github.com/apache/flink/pull/4883#discussion_r182155763 --- Diff: docs/dev/stream/state/checkpointing.md --- @@ -118,6 +120,9 @@ env.getCheckpointConfig.setMinPauseBetweenCheckpoints(500) // checkpoints have to complete within one minute, or are discarded env.getCheckpointConfig.setCheckpointTimeout(60000) +// prevent the tasks from failing if an error happens in their checkpointing, the checkpoint will just be declined. +env.getCheckpointConfig.setFailTasksOnCheckpointingErrors(false) --- End diff -- This line is missing from the Java tab.