Item 212
git_comments:

git_commits:

1. **summary:** [SPARK-19617][SS] Fix the race condition when starting and stopping a query quickly (branch-2.1)
   message: [SPARK-19617][SS] Fix the race condition when starting and stopping a query quickly (branch-2.1) ## What changes were proposed in this pull request? Backport #16947 to branch 2.1. Note: we still need to support old Hadoop versions in 2.1.*. ## How was this patch tested? Jenkins Author: Shixiong Zhu <shixiong@databricks.com> Closes #16979 from zsxwing/SPARK-19617-branch-2.1.

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-19617][SS]Fix the race condition when starting and stopping a query quickly
   body: ## What changes were proposed in this pull request? The streaming thread in StreamExecution uses the following ways to check if it should exit: - Catch an InterruptException. - `StreamExecution.state` is TERMINATED. When starting and stopping a query quickly, the above two checks may both fail: - Hit [HADOOP-14084](https://issues.apache.org/jira/browse/HADOOP-14084) and swallow InterruptException - StreamExecution.stop is called before `state` becomes `ACTIVE`. Then [runBatches] (https://github.com/apache/spark/blob/dcc2d540a53f0bd04baead43fdee1c170ef2b9f3/sql/core/src/main/scala/org/apache/spark/sql/execution/streaming/StreamExecu changes the state from `TERMINATED` to `ACTIVE`. If the above cases both happen, the query will hang forever. This PR changes `state` to `AtomicReference` and uses`compareAndSet` to make sure we only change the state from `INITIALIZING` to `ACTIVE`. It also removes the `runUninterruptibly` hack from ``HDFSMetadata`, because HADOOP-14084 won't cause any problem after we fix the race condition. ## How was this patch tested? Jenkins
2. **title:** [SPARK-19617][SS]Fix the race condition when starting and stopping a query quickly
   body: ## What changes were proposed in this pull request? The streaming thread in StreamExecution uses the following ways to check if it should exit: - Catch an InterruptException. - `StreamExecution.state` is TERMINATED. When starting and stopping a query quickly, the above two checks may both fail: - Hit [HADOOP-14084](https://issues.apache.org/jira/browse/HADOOP-14084) and swallow InterruptException - StreamExecution.stop is called before `state` becomes `ACTIVE`. Then [runBatches] (https://github.com/apache/spark/blob/dcc2d540a53f0bd04baead43fdee1c170ef2b9f3/sql/core/src/main/scala/org/apache/spark/sql/execution/streaming/StreamExecu changes the state from `TERMINATED` to `ACTIVE`. If the above cases both happen, the query will hang forever. This PR changes `state` to `AtomicReference` and uses`compareAndSet` to make sure we only change the state from `INITIALIZING` to `ACTIVE`. It also removes the `runUninterruptibly` hack from ``HDFSMetadata`, because HADOOP-14084 won't cause any problem after we fix the race condition. ## How was this patch tested? Jenkins
3. **title:** [SPARK-19617][SS]Fix the race condition when starting and stopping a query quickly
   body: ## What changes were proposed in this pull request? The streaming thread in StreamExecution uses the following ways to check if it should exit: - Catch an InterruptException. - `StreamExecution.state` is TERMINATED. When starting and stopping a query quickly, the above two checks may both fail: - Hit [HADOOP-14084](https://issues.apache.org/jira/browse/HADOOP-14084) and swallow InterruptException - StreamExecution.stop is called before `state` becomes `ACTIVE`. Then [runBatches] (https://github.com/apache/spark/blob/dcc2d540a53f0bd04baead43fdee1c170ef2b9f3/sql/core/src/main/scala/org/apache/spark/sql/execution/streaming/StreamExecu changes the state from `TERMINATED` to `ACTIVE`. If the above cases both happen, the query will hang forever. This PR changes `state` to `AtomicReference` and uses`compareAndSet` to make sure we only change the state from `INITIALIZING` to `ACTIVE`. It also removes the `runUninterruptibly` hack from ``HDFSMetadata`, because HADOOP-14084 won't cause any problem after we fix the race condition. ## How was this patch tested? Jenkins
4. **title:** [SPARK-19617][SS]Fix the race condition when starting and stopping a query quickly
   body: ## What changes were proposed in this pull request? The streaming thread in StreamExecution uses the following ways to check if it should exit: - Catch an InterruptException. - `StreamExecution.state` is TERMINATED. When starting and stopping a query quickly, the above two checks may both fail: - Hit [HADOOP-14084](https://issues.apache.org/jira/browse/HADOOP-14084) and swallow InterruptException - StreamExecution.stop is called before `state` becomes `ACTIVE`. Then [runBatches] (https://github.com/apache/spark/blob/dcc2d540a53f0bd04baead43fdee1c170ef2b9f3/sql/core/src/main/scala/org/apache/spark/sql/execution/streaming/StreamExecu changes the state from `TERMINATED` to `ACTIVE`. If the above cases both happen, the query will hang forever. This PR changes `state` to `AtomicReference` and uses`compareAndSet` to make sure we only change the state from `INITIALIZING` to `ACTIVE`. It also removes the `runUninterruptibly` hack from ``HDFSMetadata`, because HADOOP-14084 won't cause any problem after we fix the race condition. ## How was this patch tested? Jenkins
5. **title:** [SPARK-19617][SS]Fix the race condition when starting and stopping a query quickly
   body: ## What changes were proposed in this pull request? The streaming thread in StreamExecution uses the following ways to check if it should exit: - Catch an InterruptException. - `StreamExecution.state` is TERMINATED. When starting and stopping a query quickly, the above two checks may both fail: - Hit [HADOOP-14084](https://issues.apache.org/jira/browse/HADOOP-14084) and swallow InterruptException - StreamExecution.stop is called before `state` becomes `ACTIVE`. Then [runBatches] (https://github.com/apache/spark/blob/dcc2d540a53f0bd04baead43fdee1c170ef2b9f3/sql/core/src/main/scala/org/apache/spark/sql/execution/streaming/StreamExecu changes the state from `TERMINATED` to `ACTIVE`. If the above cases both happen, the query will hang forever. This PR changes `state` to `AtomicReference` and uses`compareAndSet` to make sure we only change the state from `INITIALIZING` to `ACTIVE`. It also removes the `runUninterruptibly` hack from ``HDFSMetadata`, because HADOOP-14084 won't cause any problem after we fix the race condition. ## How was this patch tested? Jenkins
6. **title:** [SPARK-19617][SS]Fix the race condition when starting and stopping a query quickly
   body: ## What changes were proposed in this pull request? The streaming thread in StreamExecution uses the following ways to check if it should exit: - Catch an InterruptException. - `StreamExecution.state` is TERMINATED. When starting and stopping a query quickly, the above two checks may both fail: - Hit [HADOOP-14084](https://issues.apache.org/jira/browse/HADOOP-14084) and swallow InterruptException - StreamExecution.stop is called before `state` becomes `ACTIVE`. Then [runBatches] (https://github.com/apache/spark/blob/dcc2d540a53f0bd04baead43fdee1c170ef2b9f3/sql/core/src/main/scala/org/apache/spark/sql/execution/streaming/StreamExecu changes the state from `TERMINATED` to `ACTIVE`. If the above cases both happen, the query will hang forever. This PR changes `state` to `AtomicReference` and uses`compareAndSet` to make sure we only change the state from `INITIALIZING` to `ACTIVE`. It also removes the `runUninterruptibly` hack from ``HDFSMetadata`, because HADOOP-14084 won't cause any problem after we fix the race condition. ## How was this patch tested? Jenkins
7. **title:** [SPARK-19617][SS]Fix the race condition when starting and stopping a query quickly
   body: ## What changes were proposed in this pull request? The streaming thread in StreamExecution uses the following ways to check if it should exit: - Catch an InterruptException. - `StreamExecution.state` is TERMINATED. When starting and stopping a query quickly, the above two checks may both fail: - Hit [HADOOP-14084](https://issues.apache.org/jira/browse/HADOOP-14084) and swallow InterruptException - StreamExecution.stop is called before `state` becomes `ACTIVE`. Then [runBatches] (https://github.com/apache/spark/blob/dcc2d540a53f0bd04baead43fdee1c170ef2b9f3/sql/core/src/main/scala/org/apache/spark/sql/execution/streaming/StreamExecu changes the state from `TERMINATED` to `ACTIVE`. If the above cases both happen, the query will hang forever. This PR changes `state` to `AtomicReference` and uses`compareAndSet` to make sure we only change the state from `INITIALIZING` to `ACTIVE`. It also removes the `runUninterruptibly` hack from ``HDFSMetadata`, because HADOOP-14084 won't cause any problem after we fix the race condition. ## How was this patch tested? Jenkins
8. **title:** [SPARK-19617][SS]Fix the race condition when starting and stopping a query quickly
   body: ## What changes were proposed in this pull request? The streaming thread in StreamExecution uses the following ways to check if it should exit: - Catch an InterruptException. - `StreamExecution.state` is TERMINATED. When starting and stopping a query quickly, the above two checks may both fail: - Hit [HADOOP-14084](https://issues.apache.org/jira/browse/HADOOP-14084) and swallow InterruptException - StreamExecution.stop is called before `state` becomes `ACTIVE`. Then [runBatches] (https://github.com/apache/spark/blob/dcc2d540a53f0bd04baead43fdee1c170ef2b9f3/sql/core/src/main/scala/org/apache/spark/sql/execution/streaming/StreamExecu changes the state from `TERMINATED` to `ACTIVE`. If the above cases both happen, the query will hang forever. This PR changes `state` to `AtomicReference` and uses`compareAndSet` to make sure we only change the state from `INITIALIZING` to `ACTIVE`. It also removes the `runUninterruptibly` hack from ``HDFSMetadata`, because HADOOP-14084 won't cause any problem after we fix the race condition. ## How was this patch tested? Jenkins
   label: code-design

9. **title:** [SPARK-19617][SS]Fix the race condition when starting and stopping a query quickly
body: ## What changes were proposed in this pull request? The streaming thread in StreamExecution uses the following ways to check if it should exit: - Catch an InterruptException. - `StreamExecution.state` is TERMINATED. When starting and stopping a query quickly, the above two checks may both fail: - Hit [HADOOP-14084](https://issues.apache.org/jira/browse/HADOOP-14084) and swallow InterruptException - StreamExecution.stop is called before `state` becomes `ACTIVE`. Then [runBatches] (https://github.com/apache/spark/blob/dcc2d540a53f0bd04baead43fdee1c170ef2b9f3/sql/core/src/main/scala/org/apache/spark/sql/execution/streaming/StreamExecu changes the state from `TERMINATED` to `ACTIVE`. If the above cases both happen, the query will hang forever. This PR changes `state` to `AtomicReference` and uses`compareAndSet` to make sure we only change the state from `INITIALIZING` to `ACTIVE`. It also removes the `runUninterruptibly` hack from ``HDFSMetadata`, because HADOOP-14084 won't cause any problem after we fix the race condition. ## How was this patch tested? Jenkins
10. **title:** [SPARK-19617][SS]Fix the race condition when starting and stopping a query quickly
body: ## What changes were proposed in this pull request? The streaming thread in StreamExecution uses the following ways to check if it should exit: - Catch an InterruptException. - `StreamExecution.state` is TERMINATED. When starting and stopping a query quickly, the above two checks may both fail: - Hit [HADOOP-14084](https://issues.apache.org/jira/browse/HADOOP-14084) and swallow InterruptException - StreamExecution.stop is called before `state` becomes `ACTIVE`. Then [runBatches] (https://github.com/apache/spark/blob/dcc2d540a53f0bd04baead43fdee1c170ef2b9f3/sql/core/src/main/scala/org/apache/spark/sql/execution/streaming/StreamExecu changes the state from `TERMINATED` to `ACTIVE`. If the above cases both happen, the query will hang forever. This PR changes `state` to `AtomicReference` and uses`compareAndSet` to make sure we only change the state from `INITIALIZING` to `ACTIVE`. It also removes the `runUninterruptibly` hack from ``HDFSMetadata`, because HADOOP-14084 won't cause any problem after we fix the race condition. ## How was this patch tested? Jenkins
11. **title:** [SPARK-19617][SS]Fix the race condition when starting and stopping a query quickly
body: ## What changes were proposed in this pull request? The streaming thread in StreamExecution uses the following ways to check if it should exit: - Catch an InterruptException. - `StreamExecution.state` is TERMINATED. When starting and stopping a query quickly, the above two checks may both fail: - Hit [HADOOP-14084](https://issues.apache.org/jira/browse/HADOOP-14084) and swallow InterruptException - StreamExecution.stop is called before `state` becomes `ACTIVE`. Then [runBatches] (https://github.com/apache/spark/blob/dcc2d540a53f0bd04baead43fdee1c170ef2b9f3/sql/core/src/main/scala/org/apache/spark/sql/execution/streaming/StreamExecu changes the state from `TERMINATED` to `ACTIVE`. If the above cases both happen, the query will hang forever. This PR changes `state` to `AtomicReference` and uses`compareAndSet` to make sure we only change the state from `INITIALIZING` to `ACTIVE`. It also removes the `runUninterruptibly` hack from ``HDFSMetadata`, because HADOOP-14084 won't cause any problem after we fix the race condition. ## How was this patch tested? Jenkins
12. **title:** [SPARK-19617][SS]Fix the race condition when starting and stopping a query quickly
body: ## What changes were proposed in this pull request? The streaming thread in StreamExecution uses the following ways to check if it should exit: - Catch an InterruptException. - `StreamExecution.state` is TERMINATED. When starting and stopping a query quickly, the above two checks may both fail: - Hit [HADOOP-14084](https://issues.apache.org/jira/browse/HADOOP-14084) and swallow InterruptException - StreamExecution.stop is called before `state` becomes `ACTIVE`. Then [runBatches] (https://github.com/apache/spark/blob/dcc2d540a53f0bd04baead43fdee1c170ef2b9f3/sql/core/src/main/scala/org/apache/spark/sql/execution/streaming/StreamExecu changes the state from `TERMINATED` to `ACTIVE`. If the above cases both happen, the query will hang forever. This PR changes `state` to `AtomicReference` and uses`compareAndSet` to make sure we only change the state from `INITIALIZING` to `ACTIVE`. It also removes the `runUninterruptibly` hack from ``HDFSMetadata`, because HADOOP-14084 won't cause any problem after we fix the race condition. ## How was this patch tested? Jenkins
label: code-design
13. **title:** [SPARK-19617][SS]Fix the race condition when starting and stopping a query quickly
body: ## What changes were proposed in this pull request? The streaming thread in StreamExecution uses the following ways to check if it should exit: - Catch an InterruptException. - `StreamExecution.state` is TERMINATED. When starting and stopping a query quickly, the above two checks may both fail: - Hit [HADOOP-14084](https://issues.apache.org/jira/browse/HADOOP-14084) and swallow InterruptException - StreamExecution.stop is called before `state` becomes `ACTIVE`. Then [runBatches] (https://github.com/apache/spark/blob/dcc2d540a53f0bd04baead43fdee1c170ef2b9f3/sql/core/src/main/scala/org/apache/spark/sql/execution/streaming/StreamExecu changes the state from `TERMINATED` to `ACTIVE`. If the above cases both happen, the query will hang forever. This PR changes `state` to `AtomicReference` and uses`compareAndSet` to make sure we only change the state from `INITIALIZING` to `ACTIVE`. It also removes the `runUninterruptibly` hack from ``HDFSMetadata`, because HADOOP-14084 won't cause any problem after we fix the race condition. ## How was this patch tested? Jenkins

github_pulls_comments:

1. **[Test build #72972 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/72972/testReport)** for PR 16947 at commit [`d52ac13`] (https://github.com/apache/spark/commit/d52ac13252798791115930ea8db9a05797e7ed7a). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
2. **[Test build #72988 has started](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/72988/testReport)** for PR 16947 at commit [`fb27a97`] (https://github.com/apache/spark/commit/fb27a97b148d5e074227760ae83d6b2e95520ed7).
3. retest this please
4. > It also removes the runUninterruptibly hack from ``HDFSMetadata` I will submit a backport PR for 2.1 to not include this change because this is needed for 2.1 due to HADOOP-10622 (Master only support Hadoop 2.6+, which already fixed HADOOP-10622).
5. **[Test build #3575 has finished](https://amplab.cs.berkeley.edu/jenkins/job/NewSparkPullRequestBuilder/3575/testReport)** for PR 16947 at commit [`7317b0f`](https://github.com/apache/spark/commit/7317b0fe7bff83f55750d9558c2f1edc4b703a15). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
6. minor grammar issue in the comment, otherwise LGTM.
7. LGTM. Merge when tests finish to master and 2.1
8. @tdas we need another PR for 2.1 since this PR assumes Hadoop 2.6+. I'm doing it now.
9. **[Test build #73078 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/73078/testReport)** for PR 16947 at commit [`13f76f6`] (https://github.com/apache/spark/commit/13f76f63843e7cb0a82c920de3ae016299744b0b). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
10. Thanks! Merging to master.
11. #16979 is the backport for branch-2.1.

github_pulls_reviews:

1. Most changes here are space changes. You can use https://github.com/apache/spark/pull/16947/files?w=1 to review it.
2. Didnt we disable interrupt because with local files, hadoop used shell commands to do file manipulation which could hang when interrupted? Are we removing this now because that has been fixed in hadoop?
3. > Are we removing this now because that has been fixed in hadoop? Yes. We dropped the support to Hadoop 2.5 and earlier versions.
4. **body:** finish the cleanup
label: code-design

jira_issues:

1. **summary:** Shell.joinThread swallows InterruptedException
description: In "Shell.joinThread", when the user tries to interrupt the thread that runs Shell.joinThread, it will catch InterruptedException and propagate it to thread t. However, it doesn't set the interrupt state of the current thread before returning, so the user codes won't know it's already interrupted. See https://github.com/apache/hadoop/blob/9e19f758c1950cbcfcd1969461a8a910efca0767/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/util/Shell.java#L1035
2. **summary:** Shell.joinThread swallows InterruptedException
description: In "Shell.joinThread", when the user tries to interrupt the thread that runs Shell.joinThread, it will catch InterruptedException and propagate it to thread t. However, it doesn't set the interrupt state of the current thread before returning, so the user codes won't know it's already interrupted. See

https://github.com/apache/hadoop/blob/9e19f758c1950cbcfcd1969461a8a910efca0767/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/util/Shell.java#L1035
3. **summary:** Shell.joinThread swallows InterruptedException
description: In "Shell.joinThread", when the user tries to interrupt the thread that runs Shell.joinThread, it will catch InterruptedException and propagate it to thread t. However, it doesn't set the interrupt state of the current thread before returning, so the user codes won't know it's already interrupted. See https://github.com/apache/hadoop/blob/9e19f758c1950cbcfcd1969461a8a910efca0767/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/util/Shell.java#L1035
4. **summary:** Shell.joinThread swallows InterruptedException
description: In "Shell.joinThread", when the user tries to interrupt the thread that runs Shell.joinThread, it will catch InterruptedException and propagate it to thread t. However, it doesn't set the interrupt state of the current thread before returning, so the user codes won't know it's already interrupted. See https://github.com/apache/hadoop/blob/9e19f758c1950cbcfcd1969461a8a910efca0767/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/util/Shell.java#L1035
5. **summary:** Shell.joinThread swallows InterruptedException
description: In "Shell.joinThread", when the user tries to interrupt the thread that runs Shell.joinThread, it will catch InterruptedException and propagate it to thread t. However, it doesn't set the interrupt state of the current thread before returning, so the user codes won't know it's already interrupted. See https://github.com/apache/hadoop/blob/9e19f758c1950cbcfcd1969461a8a910efca0767/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/util/Shell.java#L1035
6. **summary:** Shell.joinThread swallows InterruptedException
description: In "Shell.joinThread", when the user tries to interrupt the thread that runs Shell.joinThread, it will catch InterruptedException and propagate it to thread t. However, it doesn't set the interrupt state of the current thread before returning, so the user codes won't know it's already interrupted. See https://github.com/apache/hadoop/blob/9e19f758c1950cbcfcd1969461a8a910efca0767/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/util/Shell.java#L1035
7. **summary:** Shell.joinThread swallows InterruptedException
description: In "Shell.joinThread", when the user tries to interrupt the thread that runs Shell.joinThread, it will catch InterruptedException and propagate it to thread t. However, it doesn't set the interrupt state of the current thread before returning, so the user codes won't know it's already interrupted. See https://github.com/apache/hadoop/blob/9e19f758c1950cbcfcd1969461a8a910efca0767/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/util/Shell.java#L1035
8. **summary:** Shell.joinThread swallows InterruptedException
description: In "Shell.joinThread", when the user tries to interrupt the thread that runs Shell.joinThread, it will catch InterruptedException and propagate it to thread t. However, it doesn't set the interrupt state of the current thread before returning, so the user codes won't know it's already interrupted. See https://github.com/apache/hadoop/blob/9e19f758c1950cbcfcd1969461a8a910efca0767/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/util/Shell.java#L1035
label: test
9. **summary:** Shell.joinThread swallows InterruptedException
description: In "Shell.joinThread", when the user tries to interrupt the thread that runs Shell.joinThread, it will catch InterruptedException and propagate it to thread t. However, it doesn't set the interrupt state of the current thread before returning, so the user codes won't know it's already interrupted. See https://github.com/apache/hadoop/blob/9e19f758c1950cbcfcd1969461a8a910efca0767/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/util/Shell.java#L1035
10. **summary:** Shell.joinThread swallows InterruptedException
description: In "Shell.joinThread", when the user tries to interrupt the thread that runs Shell.joinThread, it will catch InterruptedException and propagate it to thread t. However, it doesn't set the interrupt state of the current thread before returning, so the user codes won't know it's already interrupted. See https://github.com/apache/hadoop/blob/9e19f758c1950cbcfcd1969461a8a910efca0767/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/util/Shell.java#L1035

jira_issues_comments:

1. Thanks for reporting. Can you provide a patch for this?
2. Posted v0 patch.
3. I don't think {{joinThread()}} should simply throw InterruptedException immediately. We have to observe the thread t dies. Before returning, we can self interrupt.
4. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 19s{color} | {color:blue} Docker mode activated. {color} | | {color:green}+1{color} | {color:green} @author {color} | {color:green} 0m 0s{color} | {color:green} The patch does not contain any @author tags. {color} | | {color:red}-1{color} | {color:red} test4tests {color} | {color:red} 0m 0s{color} | {color:red} The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 13m 26s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 22m 11s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 40s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 1m 23s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 18s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 1m 25s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 52s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 0m 47s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 18m 8s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 18m 8s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 36s{color} | {color:green} hadoop-common-project/hadoop-common: The patch generated 0 new + 40 unchanged - 1 fixed = 40 total (was 41) {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 1m 7s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 18s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} whitespace {color} | {color:green} 0m 0s{color} | {color:green} The patch has no whitespace issues. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 1m 39s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 49s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red} 16m 33s{color} | {color:red} hadoop-common in the patch failed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 35s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 83m 2s{color} | {color:black} {color} | \\ \\ || Reason || Tests || | Failed junit tests | hadoop.fs.TestSymlinkLocalFSFileContext | | | hadoop.fs.TestSymlinkLocalFSFileSystem | | Timed out junit tests | org.apache.hadoop.fs.TestLocalFileSystem | \\ \\ || Subsystem || Report/Notes || | Docker | Image:yetus/hadoop:a9ad5d6 | | JIRA Issue | HADOOP-14084 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12860040/HADOOP-14084.000.patch | | Optional Tests | asflicense compile javac javadoc mvninstall mvnsite unit findbugs checkstyle | | uname | Linux 2103b8bb7c78 3.13.0-106-generic #153-Ubuntu SMP Tue Dec 6 15:44:32 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux | | Build tool | maven | | Personality | /testptch/hadoop/patchprocess/precommit/personality/provided.sh | | git revision | trunk / f462e1f | | Default Java | 1.8.0_121 | | findbugs | v3.0.0 | | unit | https://builds.apache.org/job/PreCommit-HADOOP-Build/11886/artifact/patchprocess/patch-unit-hadoop-common-project_hadoop-common.txt | | Test Results | https://builds.apache.org/job/PreCommit-HADOOP-Build/11886/testReport/ | | modules | C: hadoop-common-project/hadoop-common U: hadoop-common-project/hadoop-common | | Console output | https://builds.apache.org/job/PreCommit-HADOOP-Build/11886/console | | Powered by | Apache Yetus 0.5.0-SNAPSHOT http://yetus.apache.org | This message was automatically generated.
5. Posted v1. Thanks [~liuml07]
6. I think if the {{joinThread()}} is used only in {{runCommand()}} method which does not propagate the InterruptedException, perhaps we don't have to make {{joinThread()}} re-throw the exception. Self interrupt before returning seems simpler. [~szetszwo] do you find time to comment? Thanks,
7. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 20s{color} | {color:blue} Docker mode activated. {color} | | {color:green}+1{color} | {color:green} @author {color} | {color:green} 0m 0s{color} | {color:green} The patch does not contain any @author tags. {color} | | {color:red}-1{color} | {color:red} test4tests {color} | {color:red} 0m 0s{color} | {color:red} The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 14m 33s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 17m 11s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 39s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvnsite {color} |

{color:green} 1m 11s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 16s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 1m 41s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 49s{color} | {color:green} trunk passed {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 0m 47s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 15m 47s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 15m 47s{color} | {color:green} the patch passed {color} | | {color:orange}-0{color} | {color:orange} checkstyle {color} | {color:orange} 0m 36s{color} | {color:orange} hadoop-common-project/hadoop-common: The patch generated 1 new + 41 unchanged - 0 fixed = 42 total (was 41) {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 1m 9s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 17s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} whitespace {color} | {color:green} 0m 0s{color} | {color:green} The patch has no whitespace issues. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 1m 49s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 48s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red} 8m 24s{color} | {color:red} hadoop-common in the patch failed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 35s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 68m 36s{color} | {color:black} {color} | \\ \\ || Reason || Tests || | Failed junit tests | hadoop.security.TestRaceWhenRelogin | | | hadoop.security.TestKDiag | \\ \\ || Subsystem || Report/Notes || | Docker | Image:yetus/hadoop:a9ad5d6 | | JIRA Issue | HADOOP-14084 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12860942/HADOOP-14084.001.patch | | Optional Tests | asflicense compile javac javadoc mvninstall mvnsite unit findbugs checkstyle | | uname | Linux 99d30318aa09 3.13.0-106-generic #153-Ubuntu SMP Tue Dec 6 15:44:32 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux | | Build tool | maven | | Personality | /testptch/hadoop/patchprocess/precommit/personality/provided.sh | | git revision | trunk / 01aca54 | | Default Java | 1.8.0_121 | | findbugs | v3.0.0 | | checkstyle | https://builds.apache.org/job/PreCommit-HADOOP-Build/11956/artifact/patchprocess/diff-checkstyle-hadoop-common-project_hadoop-common.txt | | unit | https://builds.apache.org/job/PreCommit-HADOOP-Build/11956/artifact/patchprocess/patch-unit-hadoop-common-project_hadoop-common.txt | | Test Results | https://builds.apache.org/job/PreCommit-HADOOP-Build/11956/testReport/ | | modules | C: hadoop-common-project/hadoop-common U: hadoop-common-project/hadoop-common | | Console output | https://builds.apache.org/job/PreCommit-HADOOP-Build/11956/console | | Powered by | Apache Yetus 0.5.0-SNAPSHOT http://yetus.apache.org | This message was automatically generated.

8. **body:** We need to decide what is the expected behavior of joinThread(..) when it is interrupted. The behavior probably depends on which thread it is joining. In our case, it is joining the errThread. Then, I think re-throwing the InterruptedException is fine, i.e. something similar to HADOOP-14084.000.patch. BTW, please see if you could add a test.
   label: test