Item 23
git_comments:

git_commits:

1. **summary:** HBASE-21915 Make FileLinkInputStream implement CanUnbuffer
   message: HBASE-21915 Make FileLinkInputStream implement CanUnbuffer Signed-off-by: Wellington Chevreuil <wellington.chevreuil@gmail.com> Signed-off-by: Esteban Gutierrez <esteban@apache.org>

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
   description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
2. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
   description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
3. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
   description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
4. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
   description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a

new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
label: test

5. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
   description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
   label: code-design
6. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
   description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
7. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
   description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
8. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
   description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
9. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
   description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
10. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
    description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a

caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.

11. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
    description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
12. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
    description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
13. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
    description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
14. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
    description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
15. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
    description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a

new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.

16. **summary:** FileLink$FileLinkInputStream doesn't implement CanUnbuffer
description: FileLinkInputStream is an InputStream which handles the indirection of where the real HFile lives. This implementation is wrapped via FSDataInputStreamWrapper and is transparent when it's being used by a caller. Often, we have an FSDataInputStreamWrapper wrapping a FileLinkInputStream which wraps an FSDataInputStream. The problem is that FileLinkInputStream does not implement the \{{CanUnbuffer}} interface, which means that the underlying {{FSDataInputStream}} for the HFile the link refers to doesn't get {{unbuffer()}} called on it. This can cause an open Socket to hang around, as described in HBASE-9393. Both [~wchevreuil] and myself have run into this, each for different users. We think the commonality as to why these users saw this (but we haven't run into it on our own) is that it requires a very large snapshot to be brought into a new system. Big kudos to [~esteban] for his help in diagnosing this as well! If this analysis is accurate, it would affect all branches.
label: requirement

jira_issues_comments:

1. | (/) *{color:green}+1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 11s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:orange}-0{color} | {color:orange} test4tests {color} | {color:orange} 0m 0s{color} | {color:orange} 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:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 43s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 2m 5s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 15s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 5m 4s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 2m 45s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 38s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 34s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 2m 19s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 2m 19s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 30s{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} shadedjars {color} | {color:green} 5m 1s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 11m 23s{color} | {color:green} Patch does not cause any errors with Hadoop 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 3m 7s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 38s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green}133m 2s{color} | {color:green} hbase-server in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 26s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black}180m 16s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:b002b0b | | JIRA Issue | HBASE-21915 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12958934/HBASE-21915.001.patch | | Optional Tests | dupname asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux 193a50f4b299 4.4.0-138-generic #164~14.04.1-Ubuntu SMP Fri Oct 5 08:56:16 UTC 2018 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build/component/dev-support/hbase-personality.sh | | git revision | master / ae0198084c | | maven | version: Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T18:33:14Z) | | Default Java | 1.8.0_181 | | findbugs | v3.1.0-RC3 | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/15996/testReport/ | | Max. process+thread count | 5514 (vs. ulimit of 10000) | | modules | C: hbase-server U: hbase-server | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/15996/console | | Powered by | Apache Yetus 0.8.0 http://yetus.apache.org | This message was automatically generated.

2. Good one lads. Whats up w/ the call to tryOpen inside the unbuffer? Are we opening the stream when we do this? If fresh open, an unbuffer makes sense? Thanks.
3. Actually had similar thought as [~stack]. Sounds like we are creating a new stream just to unbuffer, leaving any possible previous existing stream open. Shouldn't we just check if there is a current non null IS instance wrapped by FileLink and call unbuffer on it?
4. **body:** {quote}Whats up w/ the call to tryOpen inside the unbuffer? Are we opening the stream when we do this? If fresh open, an unbuffer makes sense? Thanks. {quote} {quote}Sounds like we are creating a new stream just to unbuffer, leaving any possible previous existing stream open {quote} Yup! You are both totally right. This was me hacking something together to run out the door ;). Let me polish this first patch, and put some thought into a unit test (although, I'm not sure of one that wouldn't be contrived).
   label: test
5. **body:** .002 let's just go with the null-check instead of the {{tryOpen()}} like you both suggested :) I want to make some kind of test to make sure that all of our InputStreams implement {{CanUnbuffer}} to avoid this problem in the future, but I'm thinking I should just do that separately. We can fix this today/now.
   label: code-design
6. lgtm +1
7. +1
8. Thanks folks. Will commit on QA.
9. | (/) *{color:green}+1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 12s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:orange}-0{color} | {color:orange} test4tests {color} | {color:orange} 0m 0s{color} | {color:orange} 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:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 10s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 55s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 4s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 4m 5s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 2m 19s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 30s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 11s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 53s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 1m 53s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 3s{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} shadedjars {color} | {color:green} 4m 10s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 8m 46s{color} | {color:green} Patch does not cause any errors with Hadoop 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 2m 25s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 30s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green}131m 59s{color} | {color:green} hbase-server in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 27s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black}170m 12s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:b002b0b | | JIRA Issue | HBASE-21915 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12959294/HBASE-21915.002.patch | | Optional Tests | dupname asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux 59031e0797d1 4.4.0-138-generic #164-Ubuntu SMP Tue Oct 2 17:16:02 UTC 2018 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build/component/dev-support/hbase-personality.sh | | git revision | master / c578020588 | | maven | version: Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T18:33:14Z) | | Default Java | 1.8.0_181 | | findbugs | v3.1.0-RC3 | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/16039/testReport/ | | Max. process+thread count | 4896 (vs. ulimit of 10000) | | modules | C: hbase-server

U: hbase-server | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/16039/console | | Powered by | Apache Yetus 0.8.0 http://yetus.apache.org | This message was automatically generated.

10. This did not come back to branch-1.2 and branch-1.3, as the default hadoop version of 2.5.1 does not contain {{CanUnbuffer}}. To land this there, we'll require a different patch which can be handled separately, IMO.
11. Results for branch branch-2.0 [build #1364 on builds.a.o|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.0/1364/]: (x) *{color:red}-1 overall{color}* ---- details (if available): (x) {color:red}-1 general checks{color} -- For more information [see general report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.0/1364//General_Nightly_Build_Report/] (/) {color:green}+1 jdk8 hadoop2 checks{color} -- For more information [see jdk8 (hadoop2) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.0/1364//JDK8_Nightly_Build_Report_(Hadoop2)/] (/) {color:green}+1 jdk8 hadoop3 checks{color} -- For more information [see jdk8 (hadoop3) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.0/1364//JDK8_Nightly_Build_Report_(Hadoop3)/] (/) {color:green}+1 source release artifact{color} -- See build output for details.
12. Results for branch branch-2.1 [build #879 on builds.a.o|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.1/879/]: (/) *{color:green}+1 overall{color}* ---- details (if available): (/) {color:green}+1 general checks{color} -- For more information [see general report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.1/879//General_Nightly_Build_Report/] (/) {color:green}+1 jdk8 hadoop2 checks{color} -- For more information [see jdk8 (hadoop2) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.1/879//JDK8_Nightly_Build_Report_(Hadoop2)/] (/) {color:green}+1 jdk8 hadoop3 checks{color} -- For more information [see jdk8 (hadoop3) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.1/879//JDK8_Nightly_Build_Report_(Hadoop3)/] (/) {color:green}+1 source release artifact{color} -- See build output for details. (/) {color:green}+1 client integration test{color}
13. Results for branch branch-1 [build #688 on builds.a.o|https://builds.apache.org/job/HBase%20Nightly/job/branch-1/688/]: (x) *{color:red}-1 overall{color}* ---- details (if available): (x) {color:red}-1 general checks{color} -- For more information [see general report|https://builds.apache.org/job/HBase%20Nightly/job/branch-1/688//General_Nightly_Build_Report/] (x) {color:red}-1 jdk7 checks{color} -- For more information [see jdk7 report|https://builds.apache.org/job/HBase%20Nightly/job/branch-1/688//JDK7_Nightly_Build_Report/] (x) {color:red}-1 jdk8 hadoop2 checks{color} -- For more information [see jdk8 (hadoop2) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-1/688//JDK8_Nightly_Build_Report_(Hadoop2)/] (x) {color:red}-1 source release artifact{color} -- See build output for details.
14. Results for branch master [build #809 on builds.a.o|https://builds.apache.org/job/HBase%20Nightly/job/master/809/]: (x) *{color:red}-1 overall{color}* ---- details (if available): (/) {color:green}+1 general checks{color} -- For more information [see general report|https://builds.apache.org/job/HBase%20Nightly/job/master/809//General_Nightly_Build_Report/] (x) {color:red}-1 jdk8 hadoop2 checks{color} -- For more information [see jdk8 (hadoop2) report|https://builds.apache.org/job/HBase%20Nightly/job/master/809//JDK8_Nightly_Build_Report_(Hadoop2)/] (x) {color:red}-1 jdk8 hadoop3 checks{color} -- For more information [see jdk8 (hadoop3) report|https://builds.apache.org/job/HBase%20Nightly/job/master/809//JDK8_Nightly_Build_Report_(Hadoop3)/] (/) {color:green}+1 source release artifact{color} -- See build output for details. (/) {color:green}+1 client integration test{color}
15. Results for branch branch-1.4 [build #674 on builds.a.o|https://builds.apache.org/job/HBase%20Nightly/job/branch-1.4/674/]: (x) *{color:red}-1 overall{color}* ---- details (if available): (x) {color:red}-1 general checks{color} -- For more information [see general report|https://builds.apache.org/job/HBase%20Nightly/job/branch-1.4/674//General_Nightly_Build_Report/] (x) {color:red}-1 jdk7 checks{color} -- For more information [see jdk7 report|https://builds.apache.org/job/HBase%20Nightly/job/branch-1.4/674//JDK7_Nightly_Build_Report/] (x) {color:red}-1 jdk8 hadoop2 checks{color} -- For more information [see jdk8 (hadoop2) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-1.4/674//JDK8_Nightly_Build_Report_(Hadoop2)/] (/) {color:green}+1 source release artifact{color} -- See build output for details.
16. Results for branch branch-2 [build #1697 on builds.a.o|https://builds.apache.org/job/HBase%20Nightly/job/branch-2/1697/]: (x) *{color:red}-1 overall{color}* ---- details (if available): (/) {color:green}+1 general checks{color} -- For more information [see general report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2/1697//General_Nightly_Build_Report/] (x) {color:red}-1 jdk8 hadoop2 checks{color} -- For more information [see jdk8 (hadoop2) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2/1697//JDK8_Nightly_Build_Report_(Hadoop2)/] (/) {color:green}+1 jdk8 hadoop3 checks{color} -- For more information [see jdk8 (hadoop3) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-

2/1697//JDK8_Nightly_Build_Report_(Hadoop3)/] (/) {color:green}+1 source release artifact{color} -- See build output for details. (/) {color:green}+1 client integration test{color}

17. Results for branch branch-2.2 [build #54 on builds.a.o|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.2/54/]: (x) *{color:red}-1 overall{color}* ---- details (if available): (/) {color:green}+1 general checks{color} -- For more information [see general report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.2/54//General_Nightly_Build_Report/] (x) {color:red}-1 jdk8 hadoop2 checks{color} -- For more information [see jdk8 (hadoop2) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.2/54//JDK8_Nightly_Build_Report_(Hadoop2)/] (/) {color:green}+1 jdk8 hadoop3 checks{color} -- For more information [see jdk8 (hadoop3) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.2/54//JDK8_Nightly_Build_Report_(Hadoop3)/] (/) {color:green}+1 source release artifact{color} -- See build output for details. (/) {color:green}+1 client integration test{color}
18. [~elserj] I see this on 2.6.1-2.6.3 hadoop compiles... [ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.6.1:compile (default-compile) on project hbase-server: Compilation failure: Compilation failure: [ERROR] /testptch/hbase/hbase-server/src/main/java/org/apache/hadoop/hbase/io/FileLink.java:[34,28] cannot find symbol [ERROR] symbol: class CanUnbuffer [ERROR] location: package org.apache.hadoop.fs [ERROR] /testptch/hbase/hbase-server/src/main/java/org/apache/hadoop/hbase/io/FileLink.java:[106,83] cannot find symbol [ERROR] symbol: class CanUnbuffer [ERROR] location: class org.apache.hadoop.hbase.io.FileLink [ERROR] /testptch/hbase/hbase-server/src/main/java/org/apache/hadoop/hbase/io/FileLink.java:[285,5] method does not override or implement a method from a supertype [ERROR] /testptch/hbase/hbase-server/src/main/java/org/apache/hadoop/hbase/io/FileLink.java:[290,9] cannot find symbol [ERROR] symbol: method unbuffer() [ERROR] location: variable in of type org.apache.hadoop.fs.FSDataInputStream [ERROR] -> [Help 1] See https://builds.apache.org/view/H-L/view/HBase/job/HBase%20Nightly/job/branch-2.0/1398/artifact/output-general/patch-javac-2.6.1.txt My fault. Should have caught it earlier. Was thinking of reverting this patch from branch-2.0 or what you think? Change our matrix to drop 2.6.1-2.6.3 hadoop? Thanks.
19. Let me reopen while figuring out the hadoop-2.6.1-2.6.3 issue...
20. {quote}Was thinking of reverting this patch from branch-2.0 or what you think? Change our matrix to drop 2.6.1-2.6.3 hadoop? Thanks. {quote} Argh! Hadoop, the present that keeps on giving :). Yeah, I didn't realize that I'd have to do this via reflection. I have zero qualms in dropping those old Hadoop versions (I hope no one is actually still on any of these). I am also OK if you want to revert for now because I don't have the cycles to drop current stuff and fix it.
21. Re-resolving. Will solve the branch-2.0/hadoop-2.6 issue in subtask HBASE-21988. Thanks [~elserj]
22. Results for branch branch-2.0 [build #1400 on builds.a.o|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.0/1400/]: (x) *{color:red}-1 overall{color}* ---- details (if available): (/) {color:green}+1 general checks{color} -- For more information [see general report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.0/1400//General_Nightly_Build_Report/] (x) {color:red}-1 jdk8 hadoop2 checks{color} -- For more information [see jdk8 (hadoop2) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.0/1400//JDK8_Nightly_Build_Report_(Hadoop2)/] (x) {color:red}-1 jdk8 hadoop3 checks{color} -- For more information [see jdk8 (hadoop3) report|https://builds.apache.org/job/HBase%20Nightly/job/branch-2.0/1400//JDK8_Nightly_Build_Report_(Hadoop3)/] (/) {color:green}+1 source release artifact{color} -- See build output for details.