Item 166
git_comments:

1. If confirmMutation is called out of order, discard mutations until id is reached.
2. * * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
3. Unimplemented.
4. * * A default implementation of {@link YarnConfigurationStore}. Doesn't offer * persistent configuration storage, just stores the configuration in memory.
5. * * Get a list of confirmed configuration mutations starting from a given id. * @param fromId id from which to start getting mutations, inclusive * @return list of configuration mutations
6. * * Get key-value configuration updates. * @return map of configuration updates
7. * * Get user who requested configuration change. * @return user who requested configuration change
8. * * Logs the configuration change to backing store. Generates an id associated * with this mutation, sets it in {@code logMutation}, and returns it. * @param logMutation configuration change to be persisted in write ahead log * @return id which configuration store associates with this mutation
9. * * Should be called after {@code logMutation}. Gets the pending mutation * associated with {@code id} and marks the mutation as persisted (no longer * pending). If isValid is true, merge the mutation with the persisted * configuration. * * If {@code confirmMutation} is called with ids in a different order than * was returned by {@code logMutation}, the result is implementation * dependent. * @param id id of mutation to be confirmed * @param isValid if true, update persisted configuration with mutation * associated with {@code id}. * @return true on success
10. * * Create log mutation prior to logging. * @param updates key-value configuration updates * @param user user who requested configuration change
11. * * Retrieve the persisted configuration. * @return configuration as key-value
12. * * Set transaction id of this configuration change. * @param id transaction id
13. * * Create log mutation for recovery. * @param updates key-value configuration updates * @param user user who requested configuration change * @param id transaction id of configuration change
14. * * Get the list of pending mutations, in the order they were logged. * @return list of mutations
15. * * Initialize the configuration store. * @param conf configuration to initialize store with * @param schedConf Initial key-value configuration to persist
16. * * LogMutation encapsulates the fields needed for configuration mutation * audit logging and recovery.
17. * * 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.
18. * * YarnConfigurationStore exposes the methods needed for retrieving and * persisting {@link CapacityScheduler} configuration via key-value * using write-ahead logging. When configuration mutation is requested, caller * should first log it with {@code logMutation}, which persists this pending * mutation. This mutation is merged to the persisted configuration only after * {@code confirmMutation} is called. * * On startup/recovery, caller should call {@code retrieve} to get all * confirmed mutations, then get pending mutations which were not confirmed via * {@code getPendingMutations}, and replay/confirm them via * {@code confirmMutation} as in the normal case.
19. * * Get transaction id of this configuration change. * @return transaction id
20. * * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.

git_commits:

1. **summary:** YARN-5946: Create YarnConfigurationStore interface and
   message: YARN-5946: Create YarnConfigurationStore interface and InMemoryConfigurationStore class. Contributed by Jonathan Hung

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
   description: This class provides the interface to persist YARN configurations in a backing store.
2. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
   description: This class provides the interface to persist YARN configurations in a backing store.
   label: code-design

3. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
   description: This class provides the interface to persist YARN configurations in a backing store.
   label: code-design
4. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
   description: This class provides the interface to persist YARN configurations in a backing store.
5. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
   description: This class provides the interface to persist YARN configurations in a backing store.
6. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
   description: This class provides the interface to persist YARN configurations in a backing store.
7. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
   description: This class provides the interface to persist YARN configurations in a backing store.
8. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
   description: This class provides the interface to persist YARN configurations in a backing store.
9. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
   description: This class provides the interface to persist YARN configurations in a backing store.
10. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
    description: This class provides the interface to persist YARN configurations in a backing store.
11. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
    description: This class provides the interface to persist YARN configurations in a backing store.
12. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
    description: This class provides the interface to persist YARN configurations in a backing store.
13. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
    description: This class provides the interface to persist YARN configurations in a backing store.
14. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
    description: This class provides the interface to persist YARN configurations in a backing store.
15. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
    description: This class provides the interface to persist YARN configurations in a backing store.
16. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
    description: This class provides the interface to persist YARN configurations in a backing store.
17. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
    description: This class provides the interface to persist YARN configurations in a backing store.
18. **summary:** Create YarnConfigurationStore interface and InMemoryConfigurationStore class
    description: This class provides the interface to persist YARN configurations in a backing store.

jira_issues_comments:

1. Attached an interface which supports retrieval and storage of bulk configuration changes.
2. **body:** Attached 002 patch. {{initialize}}, {{retrieve}}, and {{store}} are for retrieving/storing the configuration. {{readPersistedId}}, {{getMutations}}, and {{logMutations}} are for write ahead logging which implementations should support. This delegates the recovery part of write ahead logging to the calling class, in this case {{MutableConfigurationManager}}. This is necessary because the store does not know whether a logged configuration change is valid or not. If the calling class determines the logged mutations are valid, it can {{store}} them.
   label: code-design
3. **body:** Thanks, [~jhung]. I have a few comments. First, on the pedantic side, you're missing parameter descriptions in the {{initialize()}} and {{logMutations()}} javadocs, and you're missing the {{LogItem}} javadocs altogether. {{readPersistedId()}} returns the oldest change that hasn't been persisted--maybe call it {{getNextIdToPersist()}}? {{retrieve()}} returns a map of strings, but wouldn't it be more useful to return a {{Configuration}}? At a higher level, I don't quite get this interface. According to the diagrams on YARN-5734, all access to the conf store will go through the {{MutableConfigurationManager}}. That means that the MCM will be the one that's calling the retrieve/store half of the API and also be the one calling the WAL part of the API. If that's the case, why do you need both in the interface? Seems like the MCM should be able to handle validating the changes before storing them.
   label: code-design
4. Thanks [~templedf] for the comments. For the second part, not sure I follow. Right now the workflow from MCM perspective will be: logMutations(), refresh the scheduler with the new configuration, then store() if refreshed successfully. If we only have a store API which combines the logging + configuration change then the store will not know if it should actually persist the change or not.
5. Discussed offline with [~xgong] and [~leftnoteasy], seems there are some concerns about how this will be handled in the RM HA case. For the derby implementation, we can have a table1 containing the "last good" configuration, a table2 containing the mutations (one per row, basically an audit log), and a table3 with the "last good" transaction id (initialized at 0). Each mutation row in the audit log table will have a distinct and incrementing id. For a mutation, the workflow will be: # Client issues configuration mutation # Mutation passed to capacity scheduler, which then passes it to MCM # MCM logs the change in table2 (atomically, using derby's commit) via the YarnConfigurationStore interface. This change will have txn id one greater than whatever is in table3. # MCM calls cs.reinitialize with the in-memory CS configuration with the mutation applied # If success, MCM stores the mutation in table1 and increments the txn id in table3 (both of these are done together atomically). If failure, MCM increments the txn id in table3 but doesn't change table1. Steps 3-5 should be synchronized to prevent mutations to be logged in a different order than they are applied. For the failover case, first we can NFS mount the derby DB on both RMs. Now if there is failover anytime after step 3, the new RM will read table3, see if there are any logs in table2 with txn id greater than this value (via YarnConfigurationStore interface), and apply these mutations, calling cs.reinitialize on the current configuration with this mutation applied, and storing if valid. Note that since steps 3-5 are synchronized, there should be at most one log line from table2 to replay onto table1 when recovering. Let me know if there are any concerns with this.
6. Thanks [~jhung] for writing this up, it is very clear to me. One thing to confirm: bq. and a table3 with the "last good" transaction id (initialized at 0) It is actually means "last confirmed" transaction id, correct? I found in the step 5 it get increased even if update failed. And one minor suggestion to the data persisted: bq. If success, MCM stores the mutation in table1 and increments the txn id in table3 (both of these are done together atomically) I think derby may support this, but I'm not sure if this is common to different storage (for example, atomically update 2 HDFS file, or 2 ZK node, etc.). So I suggest to persist a transaction-id in addition to "last good" configuration to table-1. So even if write to table3 failed, we can recover the latest config in table-1. For the API, some suggestions to hide internal implementation details: 1) Do we really want {{Collection<String> removes}} as a part of logItem? I think set a key to empty value is equivalent to remove a key, correct? I would prefer to not add the {{removes}} field. 2) Who will generate "id" for each logItem? And suggest to make it to be long instead of int. 3) YarnConfigurationStore#retrieve, does it mean get from table-1 or get from table-1/2/3

(which described by your "for the failover case ..." in your previous comment)? I would prefer the latter one. 4) readPersistedId/getMutations look like internal implementation to me. Is it better to update them to {{List<LogMutation> getPendingMutations(void)}}? In summary, I think following APIs will be sufficient: {code} 1) initialize(Configuration conf, Map<String, String> schedConf); 2) retrieveLatestConfirmedConf which returns latest *good* configuration. This will be called when recovery 3) retrieveLatestConf which returns latest *not yet confirmed* configuration, this will be used by scheduler to try reinitialize. 4) logMutation to save the new mutation, and {{retrieveLatestConf}} can get updated accordingly. 5) confirmMutation(long id), to confirm the mutation, and {{retrieveLatestConfirmedConf}} can get updated accordingly. 6) List<LogMutation> getPendingMutations(void), this will be called when recovery 7) optional but may useful: List<Map<String, String>> getConfirmedConfHistory(long fromId). Admin can use this API to retrieve config history. {code} Please let me know your thoughts.

7. Thanks [~leftnoteasy] for the comments. bq. It is actually means "last confirmed" transaction id, correct? I found in the step 5 it get increased even if update failed. It is the txnid for which all logs with a lesser txnid do not need to be replayed on recovery. Either this means the log has been persisted to the store in case of successful refresh, or the mutation has been deemed invalid in case of failure to refresh (which is why it is incremented even if update fails). So in this case perhaps confirmMutation(long id) should be confirmMutation(long id, boolean isValid). bq. So I suggest to persist a transaction-id in addition to "last good" configuration to table-1. Sure, I think this is implementation dependent, in general though we can have a configuration entry with key="transaction.id" or something similar. bq. Who will generate "id" for each logItem? I think the YarnConfigurationStore should maintain the current id and generate new ones, which are returned upon logMutation calls. So when MCM receives a mutation, it will log it, which will then return an incremented id "id", then MCM will try to refresh, and will call confirmMutation("id", true/false). Here the YarnConfigurationStore can store a map of "id" to LogMutation in memory, so it can quickly store the LogMutation into table1 if confirmMutation(id, true) is called. bq. YarnConfigurationStore#retrieve, does it mean get from table-1 or get from table-1/2/3 (which described by your "for the failover case ..." in your previous comment)? I would prefer the latter one. On failover MCM would call retrieve (which returns a "conf"), and getPendingMutations, apply each pendingMutation one by one to "conf", and confirmMutation(pendingMutation.id, true/false) if refresh is successful/unsuccessful. So YarnConfigurationStore#retrieve on its own returns from table1 which may not have all logs applied, but MCM will reconstruct the updated configuration from getPendingMutations. So not sure if retrieveLatestConf is necessary (the third API in previous comment). Since MCM stores an in memory configuration, YarnConfigurationStore#retrieve and getPendingMutations should be only called once, on failover. So my proposal is: {noformat}1) initialize(Configuration conf, Map<String, String> schedConf); 2) retrieve which returns conf stored in table1 3) long logMutation(LogMutation) returns id to save the new mutation in table2 4) confirmMutation(long id, boolean isValid) to increment txnid stored in table1, and persist the logged mutation if isValid==true 5) List<LogMutation> getPendingMutations(void) for getting unconfirmed mutations{noformat} I think we can add getConfirmedConfHistory in a later patch. If no concerns with this approach, will upload patch. Thanks!
8. [~jhung], APIs mentioned in your latest make sense to me. I suggest to keep {{List<Map<String, String>> getConfirmedConfHistory(long fromId)}} in the interface but we don't have to implement/use it as of now. The reason is, while implementing store, we can keep in mind that this is a future requirement so we can consider it while designing storage format.
9. [~leftnoteasy], sounds good. I have attached a patch (003).
10. Reviewing the patch, and submitted to Jenkins.
11. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 16s{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:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s{color} | {color:green} The patch appears to include 1 new or modified test files. {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 16m 0s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 33s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 26s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 0m 36s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 16s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 1m 15s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 24s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 0m 38s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 36s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 36s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 26s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 0m 39s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 14s{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 21s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 19s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red} 39m 25s{color} | {color:red} hadoop-yarn-server-resourcemanager in the patch failed. {color} | | {color:red}-1{color} | {color:red} asflicense {color} | {color:red} 0m 17s{color} | {color:red} The patch generated 3 ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 64m 59s{color} | {color:black} {color} | \\ \\ || Reason || Tests || | Failed junit tests | hadoop.yarn.server.resourcemanager.scheduler.fair.TestFSAppStarvation | | | hadoop.yarn.server.resourcemanager.TestRMRestart | \\ \\ || Subsystem || Report/Notes || | Docker | Image:yetus/hadoop:a9ad5d6 | | JIRA Issue | YARN-5946 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12853317/YARN-5946-YARN-5734.003.patch | | Optional Tests | asflicense compile javac javadoc mvninstall mvnsite unit findbugs checkstyle | | uname | Linux ad4c0d5d8c11 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 | YARN-5734 / 6e1a544 | | Default Java | 1.8.0_121 | | findbugs | v3.0.0 | | unit | https://builds.apache.org/job/PreCommit-YARN-Build/15031/artifact/patchprocess/patch-unit-hadoop-yarn-project_hadoop-yarn_hadoop-yarn-server_hadoop-yarn-server-resourcemanager.txt | | Test Results | https://builds.apache.org/job/PreCommit-YARN-Build/15031/testReport/ | | asflicense | https://builds.apache.org/job/PreCommit-YARN-Build/15031/artifact/patchprocess/patch-asflicense-problems.txt | | modules | C: hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager U: hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager | | Console output | https://builds.apache.org/job/PreCommit-YARN-Build/15031/console | | Powered by | Apache Yetus 0.5.0-SNAPSHOT http://yetus.apache.org | This message was automatically generated.
12. Thanks, test failures seem related to YARN-5548 and YARN-6172. Attached 004 patch for license issues.
13. | (/) *{color:green}+1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 21s{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:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s{color} | {color:green} The patch appears to include 1 new or modified test files.

{color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 18m 32s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 32s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 25s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 0m 35s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 15s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 1m 0s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 21s{color} | {color:green} YARN-5734 passed {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 0m 30s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 29s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 29s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 23s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvnsite {color} | {color:green} 0m 31s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 11s{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 4s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 18s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} unit {color} | {color:green} 39m 47s{color} | {color:green} hadoop-yarn-server-resourcemanager in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 17s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 66m 56s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Image:yetus/hadoop:a9ad5d6 | | JIRA Issue | YARN-5946 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12854034/YARN-5946-YARN-5734.004.patch | | Optional Tests | asflicense compile javac javadoc mvninstall mvnsite unit findbugs checkstyle | | uname | Linux 82ca5438c9e1 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 | YARN-5734 / 6e1a544 | | Default Java | 1.8.0_121 | | findbugs | v3.0.0 | | Test Results | https://builds.apache.org/job/PreCommit-YARN-Build/15044/testReport/ | | modules | C: hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager U: hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager | | Console output | https://builds.apache.org/job/PreCommit-YARN-Build/15044/console | | Powered by | Apache Yetus 0.5.0-SNAPSHOT http://yetus.apache.org | This message was automatically generated.

14. +1 Committing
15. Committed into YARN-5946. Thanks, Jonathan for working on this. And Thanks, wangda for review
16. Thanks Xuan and Wangda!
17. SUCCESS: Integrated in Jenkins build Hadoop-trunk-Commit #13057 (See [https://builds.apache.org/job/Hadoop-trunk-Commit/13057/]) YARN-5946: Create YarnConfigurationStore interface and (jhung: rev e3579a8c3b1dc58a38859b189973be5a2d23f730) * (add) hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/conf/InMemoryConfigurationStore.java * (add) hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/conf/TestYarnConfigurationStore.java * (add) hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/conf/YarnConfigurationStore.java