Item 35
git_comments:

git_commits:

1. **summary:** HBASE-6299 RS starts region open while fails ack to HMaster.sendRegionOpen() causes inconsistency in HMaster's region state and a series of successive problems
   message: HBASE-6299 RS starts region open while fails ack to HMaster.sendRegionOpen() causes inconsistency in HMaster's region state and a series of successive problems git-svn-id: https://svn.apache.org/repos/asf/hbase/branches/0.94@1392620 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** RS starting region open while failing ack to HMaster.sendRegionOpen() causes inconsistency in HMaster's region state and a series of successive problems
   description: 1. HMaster tries to assign a region to an RS. 2. HMaster creates a RegionState for this region and puts it into regionsInTransition. 3. In the first assign attempt, HMaster calls RS.openRegion(). The RS receives the open region request and starts to proceed, with success eventually. However, due to network problems, HMaster fails to receive the response for the openRegion() call, and the call times out. 4. HMaster attemps to assign for a second time, choosing another RS. 5. But since the HMaster's OpenedRegionHandler has been triggered by the region open of the previous RS, and the RegionState has already been removed from regionsInTransition, HMaster finds invalid and ignores the unassigned ZK node "RS_ZK_REGION_OPENING" updated by the second attempt. 6. The unassigned ZK node stays and a later unassign fails coz RS_ZK_REGION_CLOSING cannot be created. {code} 2012-06-29 07:03:38,870 DEBUG org.apache.hadoop.hbase.master.AssignmentManager: Using pre-existing plan for region CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568.; plan=hri=CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568., src=swbss-hadoop-004,60020,1340890123243, dest=swbss-hadoop-006,60020,1340890678078 2012-06-29 07:03:38,870 DEBUG org.apache.hadoop.hbase.master.AssignmentManager: Assigning region CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568. to swbss-hadoop-006,60020,1340890678078 2012-06-29 07:03:38,870 DEBUG org.apache.hadoop.hbase.master.AssignmentManager: Handling transition=M_ZK_REGION_OFFLINE, server=swbss-hadoop-002:60000, region=b713fd655fa02395496c5a6e39ddf568 2012-06-29 07:06:28,882 DEBUG org.apache.hadoop.hbase.master.AssignmentManager: Handling transition=RS_ZK_REGION_OPENING, server=swbss-hadoop-006,60020,1340890678078, region=b713fd655fa02395496c5a6e39ddf568 2012-06-29 07:06:32,291 DEBUG org.apache.hadoop.hbase.master.AssignmentManager: Handling transition=RS_ZK_REGION_OPENING, server=swbss-hadoop-006,60020,1340890678078, region=b713fd655fa02395496c5a6e39ddf568 2012-06-29 07:06:32,299 DEBUG org.apache.hadoop.hbase.master.AssignmentManager: Handling transition=RS_ZK_REGION_OPENED, server=swbss-hadoop-006,60020,1340890678078, region=b713fd655fa02395496c5a6e39ddf568 2012-06-29 07:06:32,299 DEBUG org.apache.hadoop.hbase.master.handler.OpenedRegionHandler: Handling OPENED event for CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568. from serverName=swbss-hadoop-006,60020,1340890678078, load=(requests=518945, regions=575, usedHeap=15282, maxHeap=31301); deleting unassigned node 2012-06-29 07:06:32,299 DEBUG org.apache.hadoop.hbase.zookeeper.ZKAssign: master:60000-0x2377fee2ae80007 Deleting existing unassigned node for b713fd655fa02395496c5a6e39ddf568 that is in expected state RS_ZK_REGION_OPENED 2012-06-29 07:06:32,301 DEBUG org.apache.hadoop.hbase.zookeeper.ZKAssign: master:60000-0x2377fee2ae80007 Successfully deleted unassigned node for region b713fd655fa02395496c5a6e39ddf568 in expected state RS_ZK_REGION_OPENED 2012-06-29 07:06:32,301 DEBUG org.apache.hadoop.hbase.master.handler.OpenedRegionHandler: The master has opened the region CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568. that was online on serverName=swbss-hadoop-006,60020,1340890678078, load=(requests=518945, regions=575, usedHeap=15282, maxHeap=31301) 2012-06-29 07:07:41,140 WARN org.apache.hadoop.hbase.master.AssignmentManager: Failed assignment of CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568. to serverName=swbss-hadoop-006,60020,1340890678078, load=(requests=0, regions=575, usedHeap=0, maxHeap=0), trying to assign elsewhere instead; retry=0 java.net.SocketTimeoutException: Call to /172.16.0.6:60020 failed on socket timeout exception: java.net.SocketTimeoutException: 120000 millis timeout while waiting for channel to be ready for read. ch : java.nio.channels.SocketChannel[connected local=/172.16.0.2:53765 remote=/172.16.0.6:60020] at org.apache.hadoop.hbase.ipc.HBaseClient.wrapException(HBaseClient.java:805) at org.apache.hadoop.hbase.ipc.HBaseClient.call(HBaseClient.java:778) at org.apache.hadoop.hbase.ipc.HBaseRPC$Invoker.invoke(HBaseRPC.java:283) at $Proxy8.openRegion(Unknown Source) at org.apache.hadoop.hbase.master.ServerManager.sendRegionOpen(ServerManager.java:573) at org.apache.hadoop.hbase.master.AssignmentManager.assign(AssignmentManager.java:1127) at

org.apache.hadoop.hbase.master.AssignmentManager.assign(AssignmentManager.java:912) at
org.apache.hadoop.hbase.master.AssignmentManager.assign(AssignmentManager.java:892) at
org.apache.hadoop.hbase.master.handler.ClosedRegionHandler.process(ClosedRegionHandler.java:92) at
org.apache.hadoop.hbase.executor.EventHandler.run(EventHandler.java:162) at
java.util.concurrent.ThreadPoolExecutor$Worker.runTask(ThreadPoolExecutor.java:886) at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:908) at java.lang.Thread.run(Thread.java:662)
Caused by: java.net.SocketTimeoutException: 120000 millis timeout while waiting for channel to be ready for read. ch :
java.nio.channels.SocketChannel[connected local=/172.16.0.2:53765 remote=/172.16.0.6:60020] at
org.apache.hadoop.net.SocketIOWithTimeout.doIO(SocketIOWithTimeout.java:164) at
org.apache.hadoop.net.SocketInputStream.read(SocketInputStream.java:155) at
org.apache.hadoop.net.SocketInputStream.read(SocketInputStream.java:128) at
java.io.FilterInputStream.read(FilterInputStream.java:116) at
org.apache.hadoop.hbase.ipc.HBaseClient$Connection$PingInputStream.read(HBaseClient.java:301) at
java.io.BufferedInputStream.fill(BufferedInputStream.java:218) at
java.io.BufferedInputStream.read(BufferedInputStream.java:237) at
java.io.DataInputStream.readInt(DataInputStream.java:370) at
org.apache.hadoop.hbase.ipc.HBaseClient$Connection.receiveResponse(HBaseClient.java:541) at
org.apache.hadoop.hbase.ipc.HBaseClient$Connection.run(HBaseClient.java:479) 2012-06-29 07:07:41,142 DEBUG
org.apache.hadoop.hbase.master.AssignmentManager: No previous transition plan was found (or we are ignoring an existing
plan) for CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568.
so generated a random one;
hri=CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568.,
src=, dest=swbss-hadoop-164,60020,1340888346294; 15 (online=15, exclude=serverName=swbss-hadoop-
006,60020,1340890678078, load=(requests=0, regions=575, usedHeap=0, maxHeap=0)) available servers 2012-06-29
07:07:41,142 DEBUG org.apache.hadoop.hbase.zookeeper.ZKAssign: master:60000-0x2377fee2ae80007 Creating (or
updating) unassigned node for b713fd655fa02395496c5a6e39ddf568 with OFFLINE state 2012-06-29 07:07:41,145 DEBUG
org.apache.hadoop.hbase.master.AssignmentManager: Using pre-existing plan for region
CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568.;
plan=hri=CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568.,
src=, dest=swbss-hadoop-164,60020,1340888346294 2012-06-29 07:07:41,145 DEBUG
org.apache.hadoop.hbase.master.AssignmentManager: Assigning region
CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568. to
swbss-hadoop-164,60020,1340888346294 2012-06-29 07:07:41,149 DEBUG
org.apache.hadoop.hbase.master.AssignmentManager: Handling transition=RS_ZK_REGION_OPENING, server=swbss-
hadoop-164,60020,1340888346294, region=b713fd655fa02395496c5a6e39ddf568 2012-06-29 07:07:41,150 WARN
org.apache.hadoop.hbase.master.AssignmentManager: Received OPENING for region b713fd655fa02395496c5a6e39ddf568
from server swbss-hadoop-164,60020,1340888346294 but region was in the state null and not in expected PENDING_OPEN
or OPENING states 2012-06-29 07:07:41,296 DEBUG org.apache.hadoop.hbase.master.AssignmentManager: Handling
transition=RS_ZK_REGION_OPENING, server=swbss-hadoop-164,60020,1340888346294,
region=b713fd655fa02395496c5a6e39ddf568 2012-06-29 07:07:41,296 WARN
org.apache.hadoop.hbase.master.AssignmentManager: Received OPENING for region b713fd655fa02395496c5a6e39ddf568
from server swbss-hadoop-164,60020,1340888346294 but region was in the state null and not in expected PENDING_OPEN
or OPENING states 2012-06-29 07:07:41,302 DEBUG org.apache.hadoop.hbase.master.AssignmentManager: Handling
transition=RS_ZK_REGION_OPENED, server=swbss-hadoop-164,60020,1340888346294,
region=b713fd655fa02395496c5a6e39ddf568 2012-06-29 07:07:41,302 WARN
org.apache.hadoop.hbase.master.AssignmentManager: Received OPENED for region b713fd655fa02395496c5a6e39ddf568
from server swbss-hadoop-164,60020,1340888346294 but region was in the state null and not in expected PENDING_OPEN
or OPENING states 2012-06-29 07:08:38,872 INFO org.apache.hadoop.hbase.master.HMaster: balance
hri=CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568.,
src=swbss-hadoop-006,60020,1340890678078, dest=swbss-hadoop-008,60020,1340891085175 2012-06-29 07:08:38,872
DEBUG org.apache.hadoop.hbase.master.AssignmentManager: Starting unassignment of region
CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568.
(offlining) 2012-06-29 07:08:47,875 DEBUG org.apache.hadoop.hbase.master.AssignmentManager: Sent CLOSE to
serverName=swbss-hadoop-006,60020,1340890678078, load=(requests=0, regions=0, usedHeap=0, maxHeap=0) for region
CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568. ... 2012-
06-29 08:04:37,681 INFO org.apache.hadoop.hbase.master.AssignmentManager: Regions in transition timed out:
CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568.
state=PENDING_CLOSE, ts=1340926468331, server=null 2012-06-29 08:04:37,681 INFO
org.apache.hadoop.hbase.master.AssignmentManager: Region has been PENDING_CLOSE for too long, running forced
unassign again on
region=CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568.
2012-06-29 08:04:47,681 INFO org.apache.hadoop.hbase.master.AssignmentManager: Regions in transition timed out:
CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568.
state=PENDING_CLOSE, ts=1340926468331, server=null 2012-06-29 08:04:47,682 INFO
org.apache.hadoop.hbase.master.AssignmentManager: Region has been PENDING_CLOSE for too long, running forced
unassign again on
region=CDR_STATS_TRAFFIC,13184390567|20120508|17||2|3|913,1337256975556.b713fd655fa02395496c5a6e39ddf568.
{code}

jira_issues_comments:

1. Think a good option can be checking if the region has been assigned successfully already when dealing with the RPC failure, so that there is no need to start another attempt.
2. Add handling of SocketTimeoutException in assign(). 1. return if region is already opened on this RS. 2. try assigning on the same RS again otherwise.
3. **body:** -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12534181/HBASE-6299.patch against trunk revision . +1 @author. The patch does not contain any @author tags. -1 tests included. 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. +1 hadoop2.0. The patch compiles against the hadoop 2.0 profile. +1 javadoc. The javadoc tool did not generate any warning messages. -1 javac. The applied patch generated 5 javac compiler warnings (more than the trunk's current 4 warnings). -1 findbugs. The patch appears to introduce 7 new Findbugs (version 1.3.9) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed these unit tests: org.apache.hadoop.hbase.master.TestSplitLogManager Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/2308//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2308//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-common.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2308//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-server.html Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/2308//console This message is automatically generated.
   label: code-design
4. **body:** Good stuff Maryann. Your patch looks good as bandaid till we figure means of regionservers signaling the master they are working on a region if its taking > socket timeout to open a region. I notice in the regionserver though that if we send over an open on a region its already opening, it does the following: {code} protected void checkIfRegionInTransition(HRegionInfo region, String currentAction) throws RegionAlreadyInTransitionException { byte[] encodedName = region.getEncodedNameAsBytes(); if (this.regionsInTransitionInRS.containsKey(encodedName)) { boolean openAction = this.regionsInTransitionInRS.get(encodedName); // The below exception message will be used in master. throw new RegionAlreadyInTransitionException("Received:" + currentAction + " for the region:" + region.getRegionNameAsString() + " ,which we are already trying to " + (openAction ? OPEN : CLOSE)+ "."); } } {code} i.e. it throws a RegionAlreadyInTransitionException to avoid queuing another open.... which if I follow things properly will result in: {code} LOG.info("Unable to communicate with the region server in order" + " to assign regions", e); {code} ... and our just exiting the open. The region open will just carry on until it succeeds or until the timeout monitor monitor kicks in and assigns it elsewhere. Should your patch handle this case and perhaps give out a nicer message than the above?
   label: code-design
5. stack, thank you for pointing this out. I was thinking the innermost assign() would handle RegionAlreadyInTransitionException and return, in the code block as follows: {code} if (t instanceof RemoteException) { t = ((RemoteException) t).unwrapRemoteException(); if (t instanceof RegionAlreadyInTransitionException) { String errorMsg = "Failed assignment in: " + plan.getDestination() + " due to " + t.getMessage(); LOG.error(errorMsg, t); return; } } {code} I just looked again at HRegionServer.openRegion(), and found that RegionAlreadyInTransitionException is wrapped as ServiceException: {code} } catch (RegionAlreadyInTransitionException rie) { LOG.warn("Region is already in transition", rie); if (isBulkAssign) { builder.addOpeningState(RegionOpeningState.OPENED); } else { throw new ServiceException(rie); } {code} But i don't see why in assign() HMaster does not unwrap RemoteException and then ServiceException as well. And since RegionAlreadyInTransitionException is always wrapped, i don't see at what situation the first code block will be called. I might be missing something or need a closer look?
6. @Maryann: Take a look at ServerManager.sendRegionOpen(): {code} OpenRegionResponse response = admin.openRegion(null, request); return ResponseConverter.getRegionOpeningState(response); } catch (ServiceException se) { throw ProtobufUtil.getRemoteException(se); {code}
7. Thank you, Zhihong! then i suppose the exception handling should be modified as: {code} if (t instanceof RegionAlreadyInTransitionException) { String errorMsg = "Failed assignment in: " + plan.getDestination() + " due to " + t.getMessage(); LOG.error(errorMsg, t); return; } {code}
8. What version of HBase did you use when you collected the log in the body of this JIRA ?
9. it happened on a 0.90 cluster. and i checked trunk code and assume the issue still exists.
10. **body:** It looks to me like we have same issue in trunk. Your suggested fix looks right Maryann. Put up a patch and I'll have a go at making a unit test for it.
    label: test
11. Make handling of RegionAlreadyInTransitionException work.
12. @Maryann We just checked over here in 0.94. {code} if (t instanceof RegionAlreadyInTransitionException) { String errorMsg = "Failed assignment in: " + plan.getDestination() + " due to " + t.getMessage(); LOG.error(errorMsg, t); return; } {code} This piece of code is correct. If we directly check instancof it doesn't match. Thanks..
13. bq. This piece of code is correct. If we directly check instancof it doesn't match. Thanks.. Is it correct or incorrect Ram? I'm not sure going by the above.
14. Am very sorry for not making it clear. {code} if (t instanceof RegionAlreadyInTransitionException) { String errorMsg = "Failed assignment in: " + plan.getDestination() + " due to " + t.getMessage(); LOG.error(errorMsg, t); return; } {code} The above piece of code is correct. The RegionAlreadyInTransition is of type RemoteException. So we need to unwrap it. In the current patch {code} if (t instanceof RegionAlreadyInTransitionException) { + String errorMsg = "Failed assignment in: " + plan.getDestination() + + " due to " + t.getMessage(); + LOG.error(errorMsg, t); + return; + } {code} This is done. It will not work. We just did a small verification of this.
15. **body:** -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12535045/HBASE-6299-v2.patch against trunk revision . +1 @author. The

patch does not contain any @author tags. -1 tests included. 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. +1 hadoop2.0. The patch compiles against the hadoop 2.0 profile. +1 javadoc. The javadoc tool did not generate any warning messages. -1 javac. The applied patch generated 5 javac compiler warnings (more than the trunk's current 4 warnings). -1 findbugs. The patch appears to introduce 7 new Findbugs (version 1.3.9) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed these unit tests: org.apache.hadoop.hbase.io.hfile.TestForceCacheImportantBlocks Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/2318//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2318//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-server.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2318//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-common.html Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/2318//console This message is automatically generated.
label: code-design

16. **body:** @Maryann/Stack I can tell one scenario where this patch will lead to inconsistency. In the patch {code} else { + // The destination region server is probably processing the region open, so it + // might be safer to try this region server again to avoid having two region + // servers open the same region. + LOG.warn("Call openRegion() to " + plan.getDestination() + + " has timed out when trying to assign " + region.getRegionNameAsString() + + ". Trying to assign to this region server again; retry=" + i, t); + state.update(RegionState.State.OFFLINE); + continue; + } {code} Now because the RS is already opening i tend to assign it to same RS and i update the inmemory state to OFFLINE. At that time the RS has moved the znode from OFFLINE to OPENING or OPENINIG to OPENED. Now there is a check in handleRegion {code} if (regionState == null || (!regionState.isPendingOpen() && !regionState.isOpening())) { LOG.warn("Received OPENING for region " + prettyPrintedRegionName + " from server " + data.getOrigin() + " but region was in " + " the state " + regionState + " and not " + "in expected PENDING_OPEN or OPENING states"); return; } {code} So the master skips the transition. Now any way as we are trying out the assignment to same RS, we will either get RegionAlreadyInTransistion or sometimes even ALREADY_OPENED. If i get ALREADY_OPENED we are handling it correctly by adding to this.regions. But if i get RegionAlreadyInTransistion we just skip the assign next time. Now in the RS side the region could have been made online by this time but the master is not aware of this. One more thing is {code} + else if (t instanceof java.net.SocketTimeoutException) { + if (this.regionsInTransition.get(region.getEncodedName()) == null + && plan.getDestination().equals(getRegionServerOfRegion(region))) { {code} Here the plan could be cleared on regionOnline if RIT is cleared? Ideally over in HBASE-6060 we were trying to see how good is the retry option in assign. Sometimes the retry option and SSH were causing double assignments which we were trying to solve. Here, Can we have an option to shutdown the RS incase of sockettimeout by the master so that atleast we are sure that the SSH will take care of ssignment?
label: code-design
17. Agree, ramkrishna! You've made a good point here. My original idea was to directly return in the "else" branch, and leave it to the TimeoutMonitor to assign this region if the RS did not open the region. I changed to the current version, thinking to bring the assign retrial earlier. But regarding the region in transition problem you pointed out, the original "return" solution looks better. {code} else { + // The destination region server is probably processing the region open, so it + // might be safer to try this region server again to avoid having two region + // servers open the same region. + LOG.error("Call openRegion() to " + plan.getDestination() + + " has timed out when trying to assign " + region.getRegionNameAsString() + + ".", t); + return; + } {code} And if we are considering removing the assign retry in HBASE-6060, problems like this one and the one in HBASE-5816 can be avoided. Think triggering SSH in case of SocketTimeout should be a different problem. There are several places in HMaster where we should consider whether to start SSH, but currently only RegionServerTracker will start SSH. Shall we open another JIRA entry to discuss this issue?
18. @Maryann bq.And if we are considering removing the assign retry in HBASE-6060 Assign retry was a point discussed over there, but still not concluded on removing it. bq.Shall we open another JIRA entry to discuss this issue? Yes...sure...Stack, Jon and others have started to work on issues related to Assignments recently.
19. You are right Ram. Good one. So we should not set RegionState back to OFFLINE in this case; just update the time on it... if it has not been changed. Shutdown of RS seems extreme on socket timeout. Hopefully we don't have to do that. In particular, I'm thinking socket timeout could be easy when we give regionserver a bulk assign and cluster is busy.
20. @stack but assign() checks RegionState OFFLINE at the beginning of each attempt, and not setting it OFFLINE might cause master to abort, as in HBASE-5816: {code} for (int i = 0; i < this.maximumAssignmentAttempts; i++) { int versionOfOfflineNode = -1; if (setOfflineInZK) { // get the version of the znode after setting it to OFFLINE. // versionOfOfflineNode will be -1 if the znode was not set to OFFLINE versionOfOfflineNode = setOfflineInZooKeeper(state, hijack); {code} {code} int setOfflineInZooKeeper(final RegionState state, boolean hijack) { // In case of reassignment the current state in memory need not be // OFFLINE. if (!hijack && !state.isClosed() && !state.isOffline()) { String msg = "Unexpected state : " + state + " .. Cannot transit it to OFFLINE."; this.master.abort(msg, new IllegalStateException(msg)); return -1; } {code}
21. Retries shouldn't do that code path. Dont' you think? We set it OFFLINE once. If its other than OFFLINE, then we shouldn't be processing it?
22. Yes, agree!
23. And i'm thinking to move this "setOfflineInZK" logic into forceRegionStateToOffline(). what do you think?
24. Whatever you think Maryann. I can see us checking the znode version each time through the retries after first setting it to OFFLINE and abandoning the retries if the znode version has changed on us. When do we not have the setOfflineInZK set? Seems odd not setting it OFFLINE.
25. Currently we don't check concurrent "double" assignment, while it can happen quite easily after HBASE-5396. {code} RegionState state = addToRegionsInTransition(region, hijack); synchronized (state) { assign(region, state, setOfflineInZK, forceNewPlan, hijack); } {code} We now set RegionState OFFLINE in addToRegionsInTransition(), and set ZK node OFFLINE after we get into the critical section. Why don't we set these two OFFLINE together in addToRegionsInTransition() and after getting into the critical section check if RegionState is OFFLINE? And with "double" assignment, we go directly

with assignment() without checking its current RegionState in addToRegionsInTransition() with calls forceRegionStateToOffline(). and forceRegionStateToOffline() simply "force" a RegionState Offline. {code} RegionState state = this.regionsInTransition.get(encodedName); if (state == null) { state = new RegionState(region, RegionState.State.OFFLINE); this.regionsInTransition.put(encodedName, state); } else { // If we are reassigning the node do not force in-memory state to OFFLINE. // Based on the znode state we will decide if to change in-memory state to // OFFLINE or not. It will be done before setting znode to OFFLINE state. // We often get here with state == CLOSED because ClosedRegionHandler will // assign on its tail as part of the handling of a region close. if (!hijack) { LOG.debug("Forcing OFFLINE; was=" + state); state.update(RegionState.State.OFFLINE); } } {code} With this piece of code, we normally see logs like "Forcing OFFLINE; was=regionName state=CLOSED" with load balance. but in "double" assignment, we can see "Forcing OFFLINE; was=regionName state=OPEN". Should we ensure the state is CLOSED or OFFLINE before proceeding to assignment?

26. @Mary Is HBASE-5396 committed? bq. We now set RegionState OFFLINE in addToRegionsInTransition(), and set ZK node OFFLINE after we get into the critical section. Why don't we set these two OFFLINE together in addToRegionsInTransition() and after getting into the critical section check if RegionState is OFFLINE? I see after doing an OFFLINE to RIT, we'll then synchronize on the OFFLINE RegionState. I'm not sure what synchronize does. I suppose it prevents double assign (bulk assign doesn't do any such sync)? I'm not sure what the lock on an encoded name String does in forceRegionStateToOffline does either? I'm not sure it actually works preventing our doing an forceRegionStateToOffline on same region at same time. I like the idea of trying to make the setting of in-memory OFFLINE state and zk state happen in the one place. We'd do it synchronously? Or async and not move forward w/ the assign until we got the callback? Or, should we distingush in-memory OFFLINE state from confirmed OFFLINE state that is also up in zk? I hate that 'hijack' argument. Somehow we're supposed to make these methods do the right thing but then also make it so all can be just overridden.
27. bq. I'm not sure what the lock on an encoded name String does in forceRegionStateToOffline does either? Ignore the above. This should work.
28. bq. I'm not sure what synchronize does. I suppose it prevents double assign The interesting thing is we check RegionState is OFFLINE or CLOSED before setting OFFLINE in zk and abort if the check fails; while we allow any RegionState before setting RegionState OFFLINE. And since this synchronize on RegionState does not guard the whole process (state change from PEND_OPEN to OPENED), double assignment is not prevented at all, though there's some check in setOfflineInZookeeper, but only when hijack=true. So far i've seen two error cases with double assign: 1. HBASE-5816: The second assign comes in almost at the same time with the first assignment,but gets locked by sychronized(state). After the first assignment succeeds with sendRegionOpen() and exits the synchronized block, the second assignment goes into the block and calls setOfflineInZookeeper() which fails the "RegionState Offline" check and leads to master abort. 2. The second assignment kicks in after the first assignment succeeded and deleted the ZK node but before regionOnline() is called (which removes the region from AM.regionsInTransition and adds the region to AM.regions). The second assignment starts a normal assign process, setting RegionState OFFLINE, setting ZK OFFLINE, and calls sendRegionOpen() to the same dest RS. Then, when the first assignment calls AM.regionOnline(), this region get removed from AM.regionsInTranistion. This is a double assignment to the RS. if RS chooses to cleanUpFailedOpen() as in 0.90, this region will be served nowhere and does not even exist in master's regionsInTransition; if RS chooses to proceed on with openRegion() as in trunk, master will get RS events "OPENING", "OPENED" related to NO RegionState, as in HBASE-6300. I can see we check if ZK node exists in setOfflineInZookeeper to prevent double assignment, but this check is only effective when hijack=true. Is it possible that we can do something in an earlier stage to prevent double assignment? like in forceRegionStateToOffline()? bq. @Mary Is HBASE-5396 committed? No... but explicitly calling assign() from HBaseAdmin can cause the same problem.
29. bq. Is it possible that we can do something in an earlier stage to prevent double assignment? like in forceRegionStateToOffline()? Yes. Lets try. I was going to try and write up a reproduction of the bugs you describe above in a harness so can play with them in isolation rather than have to blow up someone's world.
30. [~maryannxue] You have any updated patch for this? Can we provide one updated patch for this issue ?
31. **body:** Considering a live RS would most likely eventually get to the openRegion() request and process, it might be good just to return on SocketTimeoutException, for SocketTimeoutException indicates an uncertain state in the assign process, with potential race conditions. And this can happen if a RS is temporarily running out of IPC handlers, or if the RS's response is lost on the line.
    label: code-design
32. updated the patch as HBASE-6299-v3.patch
33. -1 overall. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12544974/HBASE-6299-v3.patch against trunk revision . +1 @author. The patch does not contain any @author tags. -1 tests included. 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. +1 hadoop2.0. The patch compiles against the hadoop 2.0 profile. +1 javadoc. The javadoc tool did not generate any warning messages. -1 javac. The patch appears to cause mvn compile goal to fail. -1 findbugs. The patch appears to cause Findbugs (version 1.3.9) to fail. +1 release audit. The applied patch does not increase the total number of release audit warnings. +1 core tests. The patch passed unit tests in . Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/2857//testReport/ Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/2857//console This message is automatically generated.
34. @Maryann Thanks for the patch. This what we just discussed over in HBASe-6438. Pls take a look at that patch also. We could actually merge both if you feel it is fine and commit them once others review it.
35. Do we still need to unwrap the exception?
36. **body:** nit: {code} + else if (t instanceof java.net.SocketTimeoutException {code} 'else' keyword is not needed above.
    label: code-design
37. @Lars the original unwrap should not work. @Ted please review the patch. @ramkrishna How about we apply this fix first and then update the patch for HBASE-6438? for as i can see HBASE-6438 is about another problem but the patch includes my old fix.

38. -1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12545092/HBASE-6299-v3.patch against trunk revision . +1 @author. The patch does not contain any @author tags. -1 tests included. 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. +1 hadoop2.0. The patch compiles against the hadoop 2.0 profile. +1 javadoc. The javadoc tool did not generate any warning messages. -1 javac. The patch appears to cause mvn compile goal to fail. -1 findbugs. The patch appears to cause Findbugs (version 1.3.9) to fail. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed these unit tests: Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/2864//testReport/ Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/2864//console This message is automatically generated.
39. **body:** @Maryann For Sockettimeout unwrap is not needed. But for RITExcepiton we may need that. As far as i have seen we need to unwrap the remote exception to know the actual exception. But SocketTimeOut does not come under that. (Reason for this i have not digged in).:)
label: code-design
40. @ramkrishna, updated the patch. misunderstood the exception handling in HBaseClient. thank you for pointing this out!
41. -1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12545102/HBASE-6299-v3.patch against trunk revision . +1 @author. The patch does not contain any @author tags. -1 tests included. 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. +1 hadoop2.0. The patch compiles against the hadoop 2.0 profile. +1 javadoc. The javadoc tool did not generate any warning messages. -1 javac. The patch appears to cause mvn compile goal to fail. -1 findbugs. The patch appears to cause Findbugs (version 1.3.9) to fail. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed these unit tests: Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/2866//testReport/ Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/2866//console This message is automatically generated.
42. @Maryann: Latest patch looks good. In the future, please increase revision number for newer patches. Thanks
43. The unwrap not being needed for Sockettimeout makes sense, since it is actually a local exception at the client (*it* timed out, rather than the server having some problem that it passed on). Anyway, no longer an issue in v3 anyway. v3 looks good. +1
44. +1 on this small, narrowly-scoped patch. [~ram_krish] Is this patch ok w/ you?
45. Yes Stack. +1 on this.
46. v3 rotted. Here is v4 which applies to trunk. Is this in the right place MaryAnn? Thanks.
47. **body:** -1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12546000/6299v4.txt against trunk revision . +1 @author. The patch does not contain any @author tags. -1 tests included. 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. +1 hadoop2.0. The patch compiles against the hadoop 2.0 profile. -1 javadoc. The javadoc tool appears to have generated 139 warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. -1 findbugs. The patch appears to introduce 7 new Findbugs (version 1.3.9) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed these unit tests: Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/2912//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2912//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-hadoop2-compat.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2912//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-server.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2912//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-hadoop1-compat.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2912//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-common.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2912//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-hadoop-compat.html Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/2912//console This message is automatically generated.
label: code-design
48. **body:** There are some hanging tests in the hadoop QA build. +1 on patch otherwise.
label: test
49. Retry
50. **body:** -1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12546076/6299v4.txt against trunk revision . +1 @author. The patch does not contain any @author tags. -1 tests included. 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. +1 hadoop2.0. The patch compiles against the hadoop 2.0 profile. -1 javadoc. The javadoc tool appears to have generated 139 warning messages. +1 javac. The applied patch does not increase the total number of javac compiler warnings. -1 findbugs. The patch appears to introduce 7 new Findbugs (version 1.3.9) warnings. +1 release audit. The applied patch does not increase the total number of release audit warnings. -1 core tests. The patch failed these unit tests: Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/2916//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2916//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-hadoop2-compat.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2916//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-server.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2916//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-hadoop1-compat.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2916//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-common.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/2916//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-

hadoop-compat.html Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/2916//console This message is automatically generated.
label: code-design

51. Here is what I committed after running all unit tests locally.
52. Committed to trunk. Thanks for the patch Maryann.
53. Committed to 0.94 too.
54. Integrated in HBase-0.94 #501 (See [https://builds.apache.org/job/HBase-0.94/501/]) HBASE-6299 RS starts region open while fails ack to HMaster.sendRegionOpen() causes inconsistency in HMaster's region state and a series of successive problems (Revision 1392620) Result = SUCCESS stack : Files : * /hbase/branches/0.94/src/main/java/org/apache/hadoop/hbase/master/AssignmentManager.java
55. Integrated in HBase-TRUNK-on-Hadoop-2.0.0 #202 (See [https://builds.apache.org/job/HBase-TRUNK-on-Hadoop-2.0.0/202/]) HBASE-6299 RS starts region open while fails ack to HMaster.sendRegionOpen() causes inconsistency in HMaster's region state and a series of successive problems (Revision 1392617) Result = FAILURE stack : Files : * /hbase/trunk/hbase-server/src/main/java/org/apache/hadoop/hbase/master/AssignmentManager.java
56. Integrated in HBase-TRUNK #3401 (See [https://builds.apache.org/job/HBase-TRUNK/3401/]) HBASE-6299 RS starts region open while fails ack to HMaster.sendRegionOpen() causes inconsistency in HMaster's region state and a series of successive problems (Revision 1392617) Result = FAILURE stack : Files : * /hbase/trunk/hbase-server/src/main/java/org/apache/hadoop/hbase/master/AssignmentManager.java
57. Integrated in HBase-0.94-security #58 (See [https://builds.apache.org/job/HBase-0.94-security/58/]) HBASE-6299 RS starts region open while fails ack to HMaster.sendRegionOpen() causes inconsistency in HMaster's region state and a series of successive problems (Revision 1392620) Result = SUCCESS stack : Files : * /hbase/branches/0.94/src/main/java/org/apache/hadoop/hbase/master/AssignmentManager.java
58. Integrated to 0.92 as well.
59. Integrated in HBase-0.92 #593 (See [https://builds.apache.org/job/HBase-0.92/593/]) HBASE-6299 RS starting region open while failing ack to HMaster.sendRegionOpen() causes inconsistency in HMaster's region state and a series of successive problems (Maryann) (Revision 1393217) Result = SUCCESS tedyu : Files : * /hbase/branches/0.92/src/main/java/org/apache/hadoop/hbase/master/AssignmentManager.java
60. Integrated in HBase-0.94-security-on-Hadoop-23 #8 (See [https://builds.apache.org/job/HBase-0.94-security-on-Hadoop-23/8/]) HBASE-6299 RS starts region open while fails ack to HMaster.sendRegionOpen() causes inconsistency in HMaster's region state and a series of successive problems (Revision 1392620) Result = FAILURE stack : Files : * /hbase/branches/0.94/src/main/java/org/apache/hadoop/hbase/master/AssignmentManager.java
61. Integrated in HBase-0.92-security #143 (See [https://builds.apache.org/job/HBase-0.92-security/143/]) HBASE-6299 RS starting region open while failing ack to HMaster.sendRegionOpen() causes inconsistency in HMaster's region state and a series of successive problems (Maryann) (Revision 1393217) Result = FAILURE tedyu : Files : * /hbase/branches/0.92/src/main/java/org/apache/hadoop/hbase/master/AssignmentManager.java
62. Fix up after bulk move overwrote some 0.94.2 fix versions w/ 0.95.0 (Noticed by Lars Hofhansl)