Item 293
git_comments:

1. * * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
2. * * This is a very specific test that verifies that if the NettyTransceiver fails * to connect it cleans up the netty channel that it has created.
3. closing the server socket will actually free up the open channel in the transceiver, which would have hung otherwise (pre AVRO-1407)
4. all that's left is Error
5. must attempt to clean up any allocated channel future

git_commits:

1. **summary:** AVRO-1407: Java: Fix infinite loop on slow connect in NettyTransceiver. Contributed by Gareth Davis.
   message: AVRO-1407: Java: Fix infinite loop on slow connect in NettyTransceiver. Contributed by Gareth Davis. git-svn-id: https://svn.apache.org/repos/asf/avro/trunk@1641894 13f79535-47bb-0310-9956-ffa450edef68 (cherry picked from commit fbaf3c399e2f34e57008d8625c76f0543a3cadf4)

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** NettyTransceiver can cause a infinite loop when slow to connect
   description: When a new {{NettyTransceiver}} is created it forces the channel to be allocated and connected to the remote host. it waits for the connectTimeout ms on the [connect channel future|https://github.com/apache/avro/blob/1579ab1ac95731630af58fc303a07c9bf28541d6/lang/java/ipc/src/main/java/org/apache/avro/ipc/NettyTransceiver.java#L this is obivously a good thing it's only that on being unsuccessful, ie {{!channelFuture.isSuccess()}} an exception is thrown and the call to the constructor fails with an {{IOException}}, but has the potential to leave a active channel associated with the {{ChannelFactory}} The problem is that a Netty {{NioClientSocketChannelFactory}} will not shutdown if there are active channels still around and if you have supplied the {{ChannelFactory}} to the {{NettyTransceiver}} then you will not be able to cancel it by calling {{ChannelFactory.releaseExternalResources()}} like the [Flume Avro RPC client does|https://github.com/apache/flume/blob/b8cf789b8509b1e5be05dd0b0b16c5d9af9698ae/flume-ng-sdk/src/main/java/org/apache/flume/api/NettyAvroRpcClient.java#L158]. In order to recreate this you need a very laggy network, where the connect attempt takes longer than the connect timeout but does actually work, this very hard to organise in a test case, although I do have a test setup using vagrant VM's that recreates this everytime, using the Flume RPC client and server. The following stack is from a production system, it won't ever leave recover until the channel is disconnected (by forcing a disconnect at the remote host) or restarting the JVM. {noformat:title=Production stack trace} "TLOG-0" daemon prio=10 tid=0x00007f581c7be800 nid=0x39a1 waiting on condition [0x00007f57ef9f2000] java.lang.Thread.State: TIMED_WAITING (parking) at sun.misc.Unsafe.park(Native Method) parking to wait for <0x00000007218b16e0> (a java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject) at java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:196) at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(AbstractQueuedSynchronizer.java:2025) at java.util.concurrent.ThreadPoolExecutor.awaitTermination(ThreadPoolExecutor.java:1253) at org.jboss.netty.util.internal.ExecutorUtil.terminate(ExecutorUtil.java:103) at org.jboss.netty.channel.socket.nio.AbstractNioWorkerPool.releaseExternalResources(AbstractNioWorkerPool.java:80) at org.jboss.netty.channel.socket.nio.NioClientSocketChannelFactory.releaseExternalResources(NioClientSocketChannelFactory.java:181) at org.apache.flume.api.NettyAvroRpcClient.connect(NettyAvroRpcClient.java:142) at org.apache.flume.api.NettyAvroRpcClient.connect(NettyAvroRpcClient.java:101) at org.apache.flume.api.NettyAvroRpcClient.configure(NettyAvroRpcClient.java:564) locked <0x00000006c30ae7b0> (a org.apache.flume.api.NettyAvroRpcClient) at org.apache.flume.api.RpcClientFactory.getInstance(RpcClientFactory.java:88) at org.apache.flume.api.LoadBalancingRpcClient.createClient(LoadBalancingRpcClient.java:214) at org.apache.flume.api.LoadBalancingRpcClient.getClient(LoadBalancingRpcClient.java:205) locked <0x00000006a97b18e8> (a org.apache.flume.api.LoadBalancingRpcClient) at org.apache.flume.api.LoadBalancingRpcClient.appendBatch(LoadBalancingRpcClient.java:95) at com.ean.platform.components.tlog.client.service.AvroRpcEventRouter$1.call(AvroRpcEventRouter.java:45) at com.ean.platform.components.tlog.client.service.AvroRpcEventRouter$1.call(AvroRpcEventRouter.java:43) {noformat} The solution is very simple, and a patch should be along in a moment.

jira_issues_comments:

1. Patch attached. I'm afraid I've been unable to create a unit test, I'm in the process of rewriting my test setup/rig to not require internal resources and be useful.
2. **body:** We were bitten by this many times with Flume OG. After many iterations of fixes to the NettyTransceiver code, we found it much easier and less buggy to simply use a Netty ChannelGroup to make sure that there are no race conditions between Channel connections and releaseExternalResources cleanup. FYI, in case you find it useful our patched code was as follows: https://github.com/frankgrimes97/flume/blob/418397479853f836d5845c4894f5a43e654628a9/flume-core/src/main/java/com/cloudera/flume/handlers/avro/AvroNettyTransceiver.java Cheers!
   label: code-design
3. Thanks.. I've being having some similar thoughts and have started hacking away at https://github.com/gid79/flume-async. I'll being having a good look at the your Transceiver.
4. Why can't the channelFuture be closed in the finally clause?
5. 10 months to respond doesn't seem too bad.... sorry. The channel only needs to be closed only on an exception, hence the catch Throwable. The core problem is that the constructor is allocating resources that can't aren't reachable if the constructor fails.
6. **body:** Is it possible to add a unit test for this?
   label: test
7. I'll have a bash now, from memory it was rather hard to recreate
8. test case that demonstrates the issue. Applying this shows the issue. It can be made to show it really clearly if you comment out the serverSocket.close() as the failing test will never terminate. applying the actual code patch fixes the test.
9. replacement for AVRO-1407-1.patch which doesn't duplicate a one of the debug statements
10. I haven't gone nuts with the test in that it only verifies the main failure case... it doesn't verify that none connect failures are propagated correctly.
11. Commit 1641894 from [~cutting] in branch 'avro/trunk' [https://svn.apache.org/r1641894] AVRO-1407: Java: Fix infinite loop on slow connect in NettyTransceiver. Contributed by Gareth Davis.
12. I committed this. Thanks, Gareth!
13. SUCCESS: Integrated in AvroJava #502 (See [https://builds.apache.org/job/AvroJava/502/]) AVRO-1407: Java: Fix infinite loop on slow connect in NettyTransceiver. Contributed by Gareth Davis. (cutting: rev 1641894) * /avro/trunk/CHANGES.txt *

/avro/trunk/lang/java/ipc/src/main/java/org/apache/avro/ipc/NettyTransceiver.java * /avro/trunk/lang/java/ipc/src/test/java/org/apache/avro/ipc/NettyTransceiverWhenFailsToConnect.java

14. [~sacharya] it seems this is an old bugfix, the fix looks compatible with 1.7 branch, and the fix version is 1.7.8. Should this also be backported to brach-1.7?
15. Yes. Feel free to cherry-pick it back to 1.7.8. I am for the time being removing the release version 1.7.8 from the Jira
16. Commit 40733f9ab971c7f3fa26bb6c1a2f6dbf88b2a612 in avro's branch refs/heads/branch-1.7 from [~cutting] [https://git-wip-us.apache.org/repos/asf?p=avro.git;h=40733f9] AVRO-1407: Java: Fix infinite loop on slow connect in NettyTransceiver. Contributed by Gareth Davis. git-svn-id: https://svn.apache.org/repos/asf/avro/trunk@1641894 13f79535-47bb-0310-9956-ffa450edef68 (cherry picked from commit fbaf3c399e2f34e57008d8625c76f0543a3cadf4)
17. Thanks Suraj, backported to branch-1.7