Item 13
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. *
3. * * 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.
4. * * This test ensures LogCaptureAppender is configured properly
5. Kick leak detection logging
6. Force GC to bring up leaks

git_commits:

1. **summary:** CAMEL-10409 Added memory leak check to camel-netty4 component
   message: CAMEL-10409 Added memory leak check to camel-netty4 component

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Double release of netty buffer
   description: It looks like CAMEL-9040 fix introduced a double release of a netty buffer that leads to IllegalReferenceCountException under load: {code} 2016-10-22 10:20:15.442 WARN 6853 --- [ClientTCPWorker] io.netty.util.ReferenceCountUtil : Failed to release a message: AdvancedLeakAwareByteBuf(PooledUnsafeDirectByteBuf(freed)) io.netty.util.IllegalReferenceCountException: refCnt: 0, decrement: 1 at io.netty.buffer.AbstractReferenceCountedByteBuf.release(AbstractReferenceCountedByteBuf.java:111) ~[netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.buffer.WrappedByteBuf.release(WrappedByteBuf.java:1029) ~[netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.buffer.AdvancedLeakAwareByteBuf.release(AdvancedLeakAwareByteBuf.java:951) ~[netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.util.ReferenceCountUtil.release(ReferenceCountUtil.java:84) ~[netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.util.ReferenceCountUtil.safeRelease(ReferenceCountUtil.java:109) ~[netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.ChannelOutboundBuffer.remove0(ChannelOutboundBuffer.java:296) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.ChannelOutboundBuffer.failFlushed(ChannelOutboundBuffer.java:621) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.AbstractChannel$AbstractUnsafe.flush0(AbstractChannel.java:869) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.nio.AbstractNioChannel$AbstractNioUnsafe.flush0(AbstractNioChannel.java:362) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.AbstractChannel$AbstractUnsafe.flush(AbstractChannel.java:823) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.DefaultChannelPipeline$HeadContext.flush(DefaultChannelPipeline.java:1296) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.AbstractChannelHandlerContext.invokeFlush0(AbstractChannelHandlerContext.java:786) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.AbstractChannelHandlerContext.invokeFlush(AbstractChannelHandlerContext.java:778) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.AbstractChannelHandlerContext.flush(AbstractChannelHandlerContext.java:759) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.CombinedChannelDuplexHandler$DelegatingChannelHandlerContext.flush(CombinedChannelDuplexHandler.java:530) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.ChannelOutboundHandlerAdapter.flush(ChannelOutboundHandlerAdapter.java:115) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.CombinedChannelDuplexHandler.flush(CombinedChannelDuplexHandler.java:355) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.AbstractChannelHandlerContext.invokeFlush0(AbstractChannelHandlerContext.java:786) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.AbstractChannelHandlerContext.invokeWriteAndFlush(AbstractChannelHandlerContext.java:812) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.AbstractChannelHandlerContext.write(AbstractChannelHandlerContext.java:824) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.AbstractChannelHandlerContext.writeAndFlush(AbstractChannelHandlerContext.java:804) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.AbstractChannelHandlerContext.writeAndFlush(AbstractChannelHandlerContext.java:841) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.DefaultChannelPipeline.writeAndFlush(DefaultChannelPipeline.java:1032) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.AbstractChannel.writeAndFlush(AbstractChannel.java:296) [netty-all-4.1.5.Final.jar:4.1.5.Final] at org.apache.camel.component.netty4.NettyHelper.writeBodyAsync(NettyHelper.java:105) [camel-netty4-2.18.0.jar:2.18.0] at org.apache.camel.component.netty4.NettyProducer.processWithConnectedChannel(NettyProducer.java:306) [camel-netty4-2.18.0.jar:2.18.0] at org.apache.camel.component.netty4.NettyProducer$ChannelConnectedListener.operationComplete(NettyProducer.java:642) [camel-netty4-2.18.0.jar:2.18.0] at org.apache.camel.component.netty4.NettyProducer$ChannelConnectedListener.operationComplete(NettyProducer.java:619) [camel-netty4-2.18.0.jar:2.18.0] at io.netty.util.concurrent.DefaultPromise.notifyListener0(DefaultPromise.java:514) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.util.concurrent.DefaultPromise.notifyListeners0(DefaultPromise.java:507) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.util.concurrent.DefaultPromise.notifyListenersNow(DefaultPromise.java:486) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.util.concurrent.DefaultPromise.notifyListeners(DefaultPromise.java:427) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.util.concurrent.DefaultPromise.trySuccess(DefaultPromise.java:111) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.DefaultChannelPromise.trySuccess(DefaultChannelPromise.java:82) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.nio.AbstractNioChannel$AbstractNioUnsafe.fulfillConnectPromise(AbstractNioChannel.java:306) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.nio.AbstractNioChannel$AbstractNioUnsafe.finishConnect(AbstractNioChannel.java:341) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.nio.NioEventLoop.processSelectedKey(NioEventLoop.java:627) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.nio.NioEventLoop.processSelectedKeysOptimized(NioEventLoop.java:551) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.nio.NioEventLoop.processSelectedKeys(NioEventLoop.java:465) [netty-all-4.1.5.Final.jar:4.1.5.Final] at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:437) [netty-all-4.1.5.Final.jar:4.1.5.Final] at

io.netty.util.concurrent.SingleThreadEventExecutor$5.run(SingleThreadEventExecutor.java:873) [netty-all-4.1.5.Final.jar:4.1.5.Final] at java.lang.Thread.run(Thread.java:745) [na:1.8.0_101] {code} The buffer is release at {code} at org.apache.camel.component.netty4.http.NettyHttpProducer$1.onDone(NettyHttpProducer.java:85) at org.apache.camel.support.SynchronizationAdapter.onComplete(SynchronizationAdapter.java:35) at org.apache.camel.util.UnitOfWorkHelper.doneSynchronizations(UnitOfWorkHelper.java:104) at org.apache.camel.impl.DefaultUnitOfWork.done(DefaultUnitOfWork.java:230) at org.apache.camel.util.UnitOfWorkHelper.doneUow(UnitOfWorkHelper.java:65) at org.apache.camel.impl.DefaultConsumer.doneUoW(DefaultConsumer.java:107) at org.apache.camel.component.netty4.handlers.ServerChannelHandler$1.done(ServerChannelHandler.java:149) at org.apache.camel.processor.CamelInternalProcessor$InternalCallback.done(CamelInternalProcessor.java:257) at org.apache.camel.processor.Pipeline$1.done(Pipeline.java:147) at org.apache.camel.processor.CamelInternalProcessor$InternalCallback.done(CamelInternalProcessor.java:257) at org.apache.camel.processor.RedeliveryErrorHandler$2.done(RedeliveryErrorHandler.java:554) at org.apache.camel.management.InstrumentationProcessor$1.done(InstrumentationProcessor.java:86) at org.apache.camel.processor.SendProcessor$1.done(SendProcessor.java:155) at org.apache.camel.component.netty4.http.NettyHttpProducer$NettyHttpProducerCallback.done(NettyHttpProducer.java:149) at org.apache.camel.component.netty4.NettyProducer$NettyProducerCallback.done(NettyProducer.java:548) at org.apache.camel.component.netty4.handlers.ClientChannelHandler.channelRead0(ClientChannelHandler.java:205) at io.netty.channel.SimpleChannelInboundHandler.channelRead(SimpleChannelInboundHandler.java:105) at io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.java:372) at io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.java:358) at io.netty.channel.AbstractChannelHandlerContext.fireChannelRead(AbstractChannelHandlerContext.java:350) at io.netty.handler.codec.MessageToMessageDecoder.channelRead(MessageToMessageDecoder.java:102) at io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.java:372) at io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.java:358) at io.netty.channel.AbstractChannelHandlerContext.fireChannelRead(AbstractChannelHandlerContext.java:350) at io.netty.channel.CombinedChannelDuplexHandler$DelegatingChannelHandlerContext.fireChannelRead(CombinedChannelDuplexHandler.java:435) at io.netty.handler.codec.ByteToMessageDecoder.fireChannelRead(ByteToMessageDecoder.java:293) at io.netty.handler.codec.ByteToMessageDecoder.channelRead(ByteToMessageDecoder.java:267) at io.netty.channel.CombinedChannelDuplexHandler.channelRead(CombinedChannelDuplexHandler.java:250) at io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.java:372) at io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.java:358) at io.netty.channel.AbstractChannelHandlerContext.fireChannelRead(AbstractChannelHandlerContext.java:350) at io.netty.channel.DefaultChannelPipeline$HeadContext.channelRead(DefaultChannelPipeline.java:1334) at io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.java:372) at io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerContext.java:358) at io.netty.channel.DefaultChannelPipeline.fireChannelRead(DefaultChannelPipeline.java:926) at io.netty.channel.nio.AbstractNioByteChannel$NioByteUnsafe.read(AbstractNioByteChannel.java:129) at io.netty.channel.nio.NioEventLoop.processSelectedKey(NioEventLoop.java:610) at io.netty.channel.nio.NioEventLoop.processSelectedKeysOptimized(NioEventLoop.java:551) at io.netty.channel.nio.NioEventLoop.processSelectedKeys(NioEventLoop.java:465) at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:437) at io.netty.util.concurrent.SingleThreadEventExecutor$5.run(SingleThreadEventExecutor.java:873) at java.lang.Thread.run(Thread.java:745) {code} and {code} at org.apache.camel.component.netty4.NettyHelper.writeBodyAsync(NettyHelper.java:105) at org.apache.camel.component.netty4.NettyProducer.processWithConnectedChannel(NettyProducer.java:306) at org.apache.camel.component.netty4.NettyProducer$ChannelConnectedListener.operationComplete(NettyProducer.java:642) at org.apache.camel.component.netty4.NettyProducer$ChannelConnectedListener.operationComplete(NettyProducer.java:619) at io.netty.util.concurrent.DefaultPromise.notifyListener0(DefaultPromise.java:514) at io.netty.util.concurrent.DefaultPromise.notifyListeners0(DefaultPromise.java:507) at io.netty.util.concurrent.DefaultPromise.notifyListenersNow(DefaultPromise.java:486) at io.netty.util.concurrent.DefaultPromise.notifyListeners(DefaultPromise.java:427) at io.netty.util.concurrent.DefaultPromise.trySuccess(DefaultPromise.java:111) at io.netty.channel.DefaultChannelPromise.trySuccess(DefaultChannelPromise.java:82) at io.netty.channel.nio.AbstractNioChannel$AbstractNioUnsafe.fulfillConnectPromise(AbstractNioChannel.java:306) at io.netty.channel.nio.AbstractNioChannel$AbstractNioUnsafe.finishConnect(AbstractNioChannel.java:341) at io.netty.channel.nio.NioEventLoop.processSelectedKey(NioEventLoop.java:627) at io.netty.channel.nio.NioEventLoop.processSelectedKeysOptimized(NioEventLoop.java:551) at io.netty.channel.nio.NioEventLoop.processSelectedKeys(NioEventLoop.java:465) at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:437) at io.netty.util.concurrent.SingleThreadEventExecutor$5.run(SingleThreadEventExecutor.java:873) at java.lang.Thread.run(Thread.java:745) {code} Note that refCnt check do not work if buffer is being used by another thread, it's just hiding a problem if it works. I will try to look into it deeper, but I am not sure if I fully understand the buffer allocation / freeing lifecycle in netty yet My test routes are: {code} from("timer:load?period=100") .to("log:client?groupInterval=5000") .to("netty4-http:http://127.0.0.1:8081/main-service"); from("netty4-http:http://0.0.0.0:8081/main-service") .to("log:main-service?groupInterval=5000") .to("netty4-http:http://127.0.0.1:8082/childService1") .to("netty4-http:http://127.0.0.1:8083/childService2"); from("netty4-http:http://0.0.0.0:8082/childService1") .to("log:childService1?groupInterval=5000") .delayer(100); from("netty4-http:http://0.0.0.0:8083/childService2") .to("log:childService2?groupInterval=5000") .delayer(100); {code}

jira_issues_comments:

1. Looked a bit into in. Note that I analyzed request handling only. For response I'd consider removing refCnt check as it does more harm (hiding a problem sometimes) My thoughts for request: * It needs to be handled in netty4 component, not netty4-http * request needs to be release only if it did not make to "write" call * I can see next options: ** Do carefull error handling properly releasing it in case of any code pathes in which request do not make up to write, don't do in synchronization ** Mark message as "released" somehow and still do in synchrnonization adapter. Note that it should work for multiple netty calls in the exchange ** Delay body creation until we are ready to send it down I prefer delay then carefull error handling. I don't really like synchronization approach for request. It's more complicated to handle (with multiple requests) and it holds buffer more than needed.
2. GitHub user tivv opened a pull request: https://github.com/apache/camel/pull/1268 CAMEL-10409: Prevent double release of request Please also merge into 2.18 as it's quite important problem that may lead to buffer corruption (buffer is being incorrectly reused) You can merge this pull request into a Git repository by running: $ git pull https://github.com/tivv/camel netty-fix Alternatively you can review and apply these changes as the patch at: https://github.com/apache/camel/pull/1268.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #1268 ---- commit 237101e76ad7750450e29fb408f35f962218c9c6 Author: Vitalii Tymchyshyn

<vit@tym.im> Date: 2016-11-14T04:51:54Z CAMEL-10409 Double release of netty buffer commit f36cb53f85692993c39c639556c4a8c540d9501f Author: Vitalii Tymchyshyn <vit@tym.im> Date: 2016-11-14T04:55:24Z CAMEL-10409 Double release of netty buffer ----

3. I've implemented careful handling in the PR
4. Thanks [~tivv] I wonder if would help with https://issues.apache.org/jira/browse/CAMEL-10301 as well - and see if you can reproduce this with latest code, and possible fix it too ;)
5. I'll see. I also see a low hanging fruit: Server must send "Connection: close" when it is going to close connection shortly (e.g. when authentication failed). Otherwise client may try to send another request over the connection being closed. Some of the tests are flacky because of this. I'll make another JIRA and fix if confirmed.
6. Applied the patch into camel-2.18.x and master branch with thanks to Vitalii.
7. There were some memory leaks in camel-netty4 unit tests after applied the patch. {code} Failed tests: NettyUDPAsyncTest>BaseNettyTest.verifyNoLeaks:89 Leaks detected while running tests: [org.apache.logging.log4j.core.impl.MutableLogEvent@7c96c85] NettyUDPMessageLargerThanDefaultBufferSizeTest>BaseNettyTest.verifyNoLeaks:89 Leaks detected while running tests: [org.apache.logging.log4j.core.impl.MutableLogEvent@73bb1337] NettyUDPObjectSyncTest>BaseNettyTest.verifyNoLeaks:89 Leaks detected while running tests: [org.apache.logging.log4j.core.impl.MutableLogEvent@5eed6dfb, org.apache.logging.log4j.core.impl.MutableLogEvent@7c96c85, org.apache.logging.log4j.core.impl.MutableLogEvent@7c96c85] NettyUDPSyncTest>BaseNettyTest.verifyNoLeaks:89 Leaks detected while running tests: [org.apache.logging.log4j.core.impl.MutableLogEvent@7c96c85, org.apache.logging.log4j.core.impl.MutableLogEvent@7c96c85] {code} I will take a look at it later today.
8. Just found this issue is cause by my old mistake in the DatagramPacketObjectEncoder. I just filled a JIRA CAMEL-10480 for it and the quick fix is on the way.
9. Applied the patch into camel master and camel-2.18.x branch with thanks to Vitalii.
10. Github user tivv closed the pull request at: https://github.com/apache/camel/pull/1268