Item 296
git_comments:

git_commits:

1. **summary:** ARTEMIS-908: Hold connection lock when issuing credits
   message: ARTEMIS-908: Hold connection lock when issuing credits

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** AMQP flow control misses unblock during heavy load
   description: I have a benchmarking setup with 1 sender and 1 receiver attached to a broker. The broker is configured with BLOCK policy when queue is full, and I've deliberately set a low global-max-size to trigger this bug. While sending messages, I get tons of these in the Artemis log, due to the flow control kicking in: 10:46:53,673 WARN [org.apache.activemq.artemis.core.server] AMQ222183: Blocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 10:46:53,674 INFO [org.apache.activemq.artemis.core.server] AMQ221046: Unblocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 However, usually after running the test for 10-30 seconds, the sender is suddenly never unblocked and I have to restart the sender to send more messages. Every time I attach the sender, I am able to send messages for 10~30 seconds until it stops. This happens with different clients, and triggers only if it can send messages really fast. After inspecting the flow control logic in the AMQP implementation, I noticed that there is a missing synchronized block when issuing credits in AMQPSessionCallback#offerProducerCredit, which is present in ProtonServerReceiverContext#flow (if sessionSPI is null). Adding a synchronized block with the connection lock to the function passed to store.checkMemory() in AMQPSessionCallback#offerProducerCredit (in the same way as in ProtonServerReceiverContext#flow), the sender seems to be unblocked correctly. I've been able to run the sender and receiver for 5 minutes without issues so far. I'll submit the patch as a github PR.
2. **summary:** AMQP flow control misses unblock during heavy load
   description: I have a benchmarking setup with 1 sender and 1 receiver attached to a broker. The broker is configured with BLOCK policy when queue is full, and I've deliberately set a low global-max-size to trigger this bug. While sending messages, I get tons of these in the Artemis log, due to the flow control kicking in: 10:46:53,673 WARN [org.apache.activemq.artemis.core.server] AMQ222183: Blocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 10:46:53,674 INFO [org.apache.activemq.artemis.core.server] AMQ221046: Unblocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 However, usually after running the test for 10-30 seconds, the sender is suddenly never unblocked and I have to restart the sender to send more messages. Every time I attach the sender, I am able to send messages for 10~30 seconds until it stops. This happens with different clients, and triggers only if it can send messages really fast. After inspecting the flow control logic in the AMQP implementation, I noticed that there is a missing synchronized block when issuing credits in AMQPSessionCallback#offerProducerCredit, which is present in ProtonServerReceiverContext#flow (if sessionSPI is null). Adding a synchronized block with the connection lock to the function passed to store.checkMemory() in AMQPSessionCallback#offerProducerCredit (in the same way as in ProtonServerReceiverContext#flow), the sender seems to be unblocked correctly. I've been able to run the sender and receiver for 5 minutes without issues so far. I'll submit the patch as a github PR.
   label: code-design
3. **summary:** AMQP flow control misses unblock during heavy load
   description: I have a benchmarking setup with 1 sender and 1 receiver attached to a broker. The broker is configured with BLOCK policy when queue is full, and I've deliberately set a low global-max-size to trigger this bug. While sending messages, I get tons of these in the Artemis log, due to the flow control kicking in: 10:46:53,673 WARN [org.apache.activemq.artemis.core.server] AMQ222183: Blocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 10:46:53,674 INFO [org.apache.activemq.artemis.core.server] AMQ221046: Unblocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 However, usually after running the test for 10-30 seconds, the sender is suddenly never unblocked and I have to restart the sender to send more messages. Every time I attach the sender, I am able to send messages for 10~30 seconds until it stops. This happens with different clients, and triggers only if it can send messages really fast. After inspecting the flow control logic in the AMQP implementation, I noticed that there is a missing synchronized block when issuing credits in AMQPSessionCallback#offerProducerCredit, which is present in ProtonServerReceiverContext#flow (if sessionSPI is null). Adding a synchronized block with the connection lock to the function passed to store.checkMemory() in AMQPSessionCallback#offerProducerCredit (in the same way as in ProtonServerReceiverContext#flow), the sender seems to be unblocked correctly. I've been able to run the sender and receiver for 5 minutes without issues so far. I'll submit the patch as a github PR.
4. **summary:** AMQP flow control misses unblock during heavy load
   description: I have a benchmarking setup with 1 sender and 1 receiver attached to a broker. The broker is configured with BLOCK policy when queue is full, and I've deliberately set a low global-max-size to trigger this bug. While sending messages, I get tons of these in the Artemis log, due to the flow control kicking in: 10:46:53,673 WARN [org.apache.activemq.artemis.core.server] AMQ222183: Blocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 10:46:53,674 INFO [org.apache.activemq.artemis.core.server] AMQ221046: Unblocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 However, usually after running the test for 10-30 seconds, the sender is suddenly never unblocked and I have to restart the sender to send more messages. Every time I attach the sender, I am able to send messages for 10~30 seconds until it stops. This happens with different clients, and triggers only if it can send messages really fast. After inspecting the flow control logic in the AMQP implementation, I noticed that there is a missing synchronized block when issuing credits in AMQPSessionCallback#offerProducerCredit, which is present in ProtonServerReceiverContext#flow (if sessionSPI is null). Adding a synchronized block with the connection lock to the function passed to store.checkMemory() in AMQPSessionCallback#offerProducerCredit (in the same way as in ProtonServerReceiverContext#flow), the sender seems to be unblocked correctly. I've been able to run the sender and receiver for 5 minutes without issues so far. I'll submit the patch as a github PR.

5. **summary:** AMQP flow control misses unblock during heavy load
   description: I have a benchmarking setup with 1 sender and 1 receiver attached to a broker. The broker is configured with BLOCK policy when queue is full, and I've deliberately set a low global-max-size to trigger this bug. While sending messages, I get tons of these in the Artemis log, due to the flow control kicking in: 10:46:53,673 WARN [org.apache.activemq.artemis.core.server] AMQ222183: Blocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 10:46:53,674 INFO [org.apache.activemq.artemis.core.server] AMQ221046: Unblocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 However, usually after running the test for 10-30 seconds, the sender is suddenly never unblocked and I have to restart the sender to send more messages. Every time I attach the sender, I am able to send messages for 10~30 seconds until it stops. This happens with different clients, and triggers only if it can send messages really fast. After inspecting the flow control logic in the AMQP implementation, I noticed that there is a missing synchronized block when issuing credits in AMQPSessionCallback#offerProducerCredit, which is present in ProtonServerReceiverContext#flow (if sessionSPI is null). Adding a synchronized block with the connection lock to the function passed to store.checkMemory() in AMQPSessionCallback#offerProducerCredit (in the same way as in ProtonServerReceiverContext#flow), the sender seems to be unblocked correctly. I've been able to run the sender and receiver for 5 minutes without issues so far. I'll submit the patch as a github PR.

6. **summary:** AMQP flow control misses unblock during heavy load
   description: I have a benchmarking setup with 1 sender and 1 receiver attached to a broker. The broker is configured with BLOCK policy when queue is full, and I've deliberately set a low global-max-size to trigger this bug. While sending messages, I get tons of these in the Artemis log, due to the flow control kicking in: 10:46:53,673 WARN [org.apache.activemq.artemis.core.server] AMQ222183: Blocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 10:46:53,674 INFO [org.apache.activemq.artemis.core.server] AMQ221046: Unblocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 However, usually after running the test for 10-30 seconds, the sender is suddenly never unblocked and I have to restart the sender to send more messages. Every time I attach the sender, I am able to send messages for 10~30 seconds until it stops. This happens with different clients, and triggers only if it can send messages really fast. After inspecting the flow control logic in the AMQP implementation, I noticed that there is a missing synchronized block when issuing credits in AMQPSessionCallback#offerProducerCredit, which is present in ProtonServerReceiverContext#flow (if sessionSPI is null). Adding a synchronized block with the connection lock to the function passed to store.checkMemory() in AMQPSessionCallback#offerProducerCredit (in the same way as in ProtonServerReceiverContext#flow), the sender seems to be unblocked correctly. I've been able to run the sender and receiver for 5 minutes without issues so far. I'll submit the patch as a github PR.
7. **summary:** AMQP flow control misses unblock during heavy load
   description: I have a benchmarking setup with 1 sender and 1 receiver attached to a broker. The broker is configured with BLOCK policy when queue is full, and I've deliberately set a low global-max-size to trigger this bug. While sending messages, I get tons of these in the Artemis log, due to the flow control kicking in: 10:46:53,673 WARN [org.apache.activemq.artemis.core.server] AMQ222183: Blocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 10:46:53,674 INFO [org.apache.activemq.artemis.core.server] AMQ221046: Unblocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 However, usually after running the test for 10-30 seconds, the sender is suddenly never unblocked and I have to restart the sender to send more messages. Every time I attach the sender, I am able to send messages for 10~30 seconds until it stops. This happens with different clients, and triggers only if it can send messages really fast. After inspecting the flow control logic in the AMQP implementation, I noticed that there is a missing synchronized block when issuing credits in AMQPSessionCallback#offerProducerCredit, which is present in ProtonServerReceiverContext#flow (if sessionSPI is null). Adding a synchronized block with the connection lock to the function passed to store.checkMemory() in AMQPSessionCallback#offerProducerCredit (in the same way as in ProtonServerReceiverContext#flow), the sender seems to be unblocked correctly. I've been able to run the sender and receiver for 5 minutes without issues so far. I'll submit the patch as a github PR.
8. **summary:** AMQP flow control misses unblock during heavy load
   description: I have a benchmarking setup with 1 sender and 1 receiver attached to a broker. The broker is configured with BLOCK policy when queue is full, and I've deliberately set a low global-max-size to trigger this bug. While sending messages, I get tons of these in the Artemis log, due to the flow control kicking in: 10:46:53,673 WARN [org.apache.activemq.artemis.core.server] AMQ222183: Blocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 10:46:53,674 INFO [org.apache.activemq.artemis.core.server] AMQ221046: Unblocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 However, usually after running the test for 10-30 seconds, the sender is suddenly never unblocked and I have to restart the sender to send more messages. Every time I attach the sender, I am able to send messages for 10~30 seconds until it stops. This happens with different clients, and triggers only if it can send messages really fast. After inspecting the flow control logic in the AMQP implementation, I noticed that there is a missing synchronized block when issuing credits in AMQPSessionCallback#offerProducerCredit, which is present in ProtonServerReceiverContext#flow (if sessionSPI is null). Adding a synchronized block with the connection lock to the function passed to store.checkMemory() in AMQPSessionCallback#offerProducerCredit (in the same way as in ProtonServerReceiverContext#flow), the sender seems to be unblocked correctly. I've been able to run the sender and receiver for 5 minutes without issues so far. I'll submit the patch as a github PR.
9. **summary:** AMQP flow control misses unblock during heavy load
   description: I have a benchmarking setup with 1 sender and 1 receiver attached to a broker. The broker is configured with BLOCK policy when queue is full, and I've deliberately set a low global-max-size to trigger this bug. While sending messages, I get tons of these in the Artemis log, due to the flow control kicking in: 10:46:53,673 WARN [org.apache.activemq.artemis.core.server] AMQ222183: Blocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 10:46:53,674 INFO [org.apache.activemq.artemis.core.server] AMQ221046: Unblocking message production on address 'myqueue'; size is currently: 105,456 bytes; max-size-bytes: -1 However, usually after running the test for 10-30 seconds, the sender is suddenly never unblocked and I have to restart the sender to send more messages. Every time I attach the sender, I am able to send messages for 10~30 seconds until it stops. This happens with different clients, and triggers only if it can send messages really fast. After inspecting the flow control logic in the AMQP implementation, I noticed that there is a missing synchronized block when issuing credits in AMQPSessionCallback#offerProducerCredit, which is present in ProtonServerReceiverContext#flow (if sessionSPI is null). Adding a synchronized block with the connection lock to the function passed to store.checkMemory() in AMQPSessionCallback#offerProducerCredit (in the same way as in ProtonServerReceiverContext#flow), the sender seems to be unblocked correctly. I've been able to run the sender and receiver for 5 minutes without issues so far. I'll submit the patch as a github PR.

jira_issues_comments:

1. GitHub user lulf opened a pull request: https://github.com/apache/activemq-artemis/pull/947 ARTEMIS-908: Hold connection lock when issuing credits You can merge this pull request into a Git repository by running: $ git pull https://github.com/EnMasseProject/activemq-artemis ARTEMIS-908 Alternatively you can review and apply these changes as the patch at: https://github.com/apache/activemq-artemis/pull/947.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 #947 ---- commit cf7040f219f316e7c2222b2a6c7b770e3cf5f7b1 Author: Ulf Lilleengen <lulf@redhat.com> Date: 2017-01-03T10:03:38Z ARTEMIS-908: Hold connection lock when issuing credits ----

2. Commit f2f355206740b6efb2359ad7296409fbfd62c5b3 in activemq-artemis's branch refs/heads/master from [~lulf] [https://git-wip-us.apache.org/repos/asf?p=activemq-artemis.git;h=f2f3552] ARTEMIS-908: Hold connection lock when issuing credits
3. Github user asfgit closed the pull request at: https://github.com/apache/activemq-artemis/pull/947
4. **body:** I see that some new code was added to the same method, but it is missing the synchronization block as well.
   label: code-design
5. GitHub user lulf opened a pull request: https://github.com/apache/activemq-artemis/pull/1016 ARTEMIS-908: Ensure that connection lock is held when flushing You can merge this pull request into a Git repository by running: $ git pull https://github.com/EnMasseProject/activemq-artemis ARTEMIS-908-reopened Alternatively you can review and apply these changes as the patch at: https://github.com/apache/activemq-artemis/pull/1016.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 #1016 ---- commit 5f6a390b6098dca1cfda09af4c0af096dfe736c0 Author: Ulf Lilleengen <lulf@redhat.com> Date: 2017-02-15T13:09:27Z ARTEMIS-908: Ensure that connection lock is held when flushing ----
6. Commit 5f6a390b6098dca1cfda09af4c0af096dfe736c0 in activemq-artemis's branch refs/heads/master from [~lulf] [https://git-wip-us.apache.org/repos/asf?p=activemq-artemis.git;h=5f6a390] ARTEMIS-908: Ensure that connection lock is held when flushing
7. Github user asfgit closed the pull request at: https://github.com/apache/activemq-artemis/pull/1016
8. Github user clebertsuconic commented on the issue: https://github.com/apache/activemq-artemis/pull/1016 I have merged this. although I remember having to open a lock like this in order to avoid dead locks.
9. After some more testing, I'm still seeing the block/unblock issue even with this patch, so seems to be some flow control issues left.. The broker says there is 1 consumer attached and 0 messages in the queue. And even if I reattach the receiver, the sender is still blocked. If I reattach the sender, it is working again until the next time the issue occurs.
10. Ok, I think I found the issue. Two threads deadlocked. Thread 37 acquires the OversizedRunnable lock and then the connection lock, while Thread 3 acquires them in opposite order. "Thread-37 (ActiveMQ-server-org.apache.activemq.artemis.core.server.impl.ActiveMQServerImpl$3@52e677af-27 4722023)" #116 prio=5 os_prio=0 tid=0x00007f386c004800 nid=0xb70a waiting for monitor entry [0x00007f389c df9000] java.lang.Thread.State: BLOCKED (on object monitor) at org.apache.activemq.artemis.protocol.amqp.broker.AMQPSessionCallback$4.run(AMQPSessionCallback .java:456) - waiting to lock <0x00000000cb54f820> (a java.lang.Object) at org.apache.activemq.artemis.core.paging.impl.PagingStoreImpl$OverSizedRunnable.run(PagingStoreImpl.java:651) - locked <0x00000000cb6ed650> (a org.apache.activemq.artemis.core.paging.impl.PagingStoreImpl$OverSizedRunnable) at org.apache.activemq.artemis.core.paging.impl.PagingStoreImpl$MemoryFreedRunnablesExecutor.run(PagingStoreImpl.java:630) at org.apache.activemq.artemis.utils.OrderedExecutorFactory$OrderedExecutor$ExecutorTask.run(OrderedExecutorFactory.java:101) at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142) at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617) at java.lang.Thread.run(Thread.java:745) "Thread-3 (ActiveMQ-server-org.apache.activemq.artemis.core.server.impl.ActiveMQServerImpl$3@52e677af-274 722023)" #62 prio=5 os_prio=0 tid=0x00007f38b8002000 nid=0xb6d1 waiting for monitor entry [0x00007f399c6e 7000] java.lang.Thread.State: BLOCKED (on object monitor) at org.apache.activemq.artemis.core.paging.impl.PagingStoreImpl$OverSizedRunnable.run(PagingStoreImpl.java:650) - waiting to lock <0x00000000cb6ed650> (a org.apache.activemq.artemis.core.paging.impl.PagingStoreImpl$OverSizedRunnable) at org.apache.activemq.artemis.core.paging.impl.PagingStoreImpl.checkMemory(PagingStoreImpl.java:677) at org.apache.activemq.artemis.protocol.amqp.broker.AMQPSessionCallback.offerProducerCredit(AMQPSessionCallback.java:453) at org.apache.activemq.artemis.protocol.amqp.proton.ProtonServerReceiverContext.flow(ProtonServerReceiverContext.java:202) at org.apache.activemq.artemis.protocol.amqp.proton.ProtonServerReceiverContext.onMessage(ProtonServerReceiverContext.java:163) - locked <0x00000000cb54f820> (a java.lang.Object) at org.apache.activemq.artemis.protocol.amqp.proton.AMQPConnectionContext$LocalListener.onDelivery(AMQPConnectionContext.java:421) at org.apache.activemq.artemis.protocol.amqp.proton.handler.Events.dispatch(Events.java:97) at org.apache.activemq.artemis.protocol.amqp.proton.handler.ProtonHandler.dispatch(ProtonHandler.java:345) at org.apache.activemq.artemis.protocol.amqp.proton.handler.ProtonHandler.access$000(ProtonHandler.java:43) at org.apache.activemq.artemis.protocol.amqp.proton.handler.ProtonHandler$1.run(ProtonHandler.java:62) at org.apache.activemq.artemis.utils.OrderedExecutorFactory$OrderedExecutor$ExecutorTask.run(OrderedExecutorFactory.java:101) at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142) at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617) at java.lang.Thread.run(Thread.java:745)
11. GitHub user lulf opened a pull request: https://github.com/apache/activemq-artemis/pull/1018 ARTEMIS-908: Replace lock by CAS to avoid potential deadlock The deadlock can happen if the Runnable acquires a lock. You can merge this pull request into a Git repository by running: $ git pull https://github.com/lulf/activemq-artemis ARTEMIS-908-cas Alternatively you can review and apply these changes as the patch at: https://github.com/apache/activemq-artemis/pull/1018.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 #1018 ---- commit 1b1e8dded9b062461a074b0e41dcc71e719c7a87 Author: Ulf Lilleengen <lulf@redhat.com> Date: 2017-02-15T14:49:11Z ARTEMIS-908: Replace lock by CAS to avoid deadlock ----
12. Commit 4622066323e9ba5af0777d0c7315389ee8f647f5 in activemq-artemis's branch refs/heads/master from [~lulf] [https://git-wip-us.apache.org/repos/asf?p=activemq-artemis.git;h=4622066] ARTEMIS-908: Replace lock by CAS to avoid deadlock
13. Github user asfgit closed the pull request at: https://github.com/apache/activemq-artemis/pull/1018
14. Commit b0585329b41fc074aea80fe952dde565dcef491d in activemq-artemis's branch refs/heads/1.x from [~lulf] [https://git-wip-us.apache.org/repos/asf?p=activemq-artemis.git;h=b058532] ARTEMIS-908: Replace lock by CAS to avoid deadlock (cherry picked from commit 4622066323e9ba5af0777d0c7315389ee8f647f5)