Item 158
git_comments:

git_commits:

1. **summary:** Update groovy (#6425)
   message: Update groovy (#6425) Fixes deadlock: https://issues.apache.org/jira/browse/GROOVY-8067

github_issues:

github_issues_comments:

github_pulls:

1. **title:** Update groovy
   body: Fixes deadlock: https://issues.apache.org/jira/browse/GROOVY-8067 ## Description Add a description of your PR here. A good description should include pointers to an issue or design document, etc. ## Upgrade Notes Does this PR prevent a zero down-time upgrade? (Assume upgrade order: Controller, Broker, Server, Minion) * [ ] Yes (Please label as **<code>backward-incompat</code>**, and complete the section below on Release Notes) Does this PR fix a zero-downtime upgrade introduced earlier? * [ ] Yes (Please label this as **<code>backward-incompat</code>**, and complete the section below on Release Notes) Does this PR otherwise need attention when creating release notes? Things to consider: - New configuration options - Deprecation of configurations - Signature changes to public methods/interfaces - New plugins added or old plugins removed * [ ] Yes (Please label this PR as **<code>release-notes</code>** and complete the section on Release Notes) ## Release Notes If you have tagged this as either backward-incompat or release-notes, you MUST add text here that you would like to see appear in release notes of the next release. If you have a series of commits adding or enabling a feature, then add this section only in final commit that marks the feature completed. Refer to earlier release notes to see examples of text ## Documentation If you have introduced a new feature or configuration, please add it to the documentation as well. See https://docs.pinot.apache.org/developers/developers-and-contributors/update-document

github_pulls_comments:

1. # [Codecov](https://codecov.io/gh/apache/incubator-pinot/pull/6425?src=pr&el=h1) Report > Merging [#6425](https://codecov.io/gh/apache/incubator-pinot/pull/6425?src=pr&el=desc) (ce187a8) into [master](https://codecov.io/gh/apache/incubator-pinot/commit/1beaab59b73f26c4e35f3b9bc856b03806cddf5a?el=desc) (1beaab5) will **decrease** coverage by `1.41%`. > The diff coverage is `56.80%`. [![Impacted file tree graph](https://codecov.io/gh/apache/incubator-pinot/pull/6425/graphs/tree.svg?width=650&height=150&src=pr&token=4ibza2ugkz)](https://codecov.io/gh/apache/incubator-pinot/pull/6425?src=pr&el=tree) ```diff @@ Coverage Diff @@ ## master #6425 +/- ## ========== - Coverage 66.44% 65.03% -1.42% ========== Files 1075 1320 +245 Lines 54773 64448 +9675 Branches 8168 9396 +1228 ========== + Hits 36396 41911 +5515 - Misses 15700 19539 +3839 - Partials 2677 2998 +321 ``` | Flag | Coverage Δ | | |---|---|---| | unittests | `65.03% <56.80%> (?)` | | Flags with carried forward coverage won't be shown. [Click here](https://docs.codecov.io/docs/carryforward-flags#carryforward-flags-in-the-pull-request-comment) to find out more. | [Impacted Files](https://codecov.io/gh/apache/incubator-pinot/pull/6425?src=pr&el=tree) | Coverage Δ | | |---|---|---| | [...e/pinot/broker/api/resources/PinotBrokerDebug.java](https://codecov.io/gh/apache/incubator-pinot/pull/6425/diff?src=pr&el=tree#diff-cGlub3QtYnJva2VyL3NyYy9tYWluL2phdmEvb3JnL2FwYWNoZS9waW5vdC9icm9rZXIvYXBpL3Jlc291cmNlcy9QaW5vdEJyb2tlckRlYnVnLmphdmE=) | `0.00% <0.00%> (-79.32%)` | :arrow_down: | | [...ot/broker/broker/AllowAllAccessControlFactory.java](https://codecov.io/gh/apache/incubator-pinot/pull/6425/diff?src=pr&el=tree#diff-cGlub3QtYnJva2VyL3NyYy9tYWluL2phdmEvb3JnL2FwYWNoZS9waW5vdC9icm9rZXIvYnJva2VyL0FsbG93QWxsQWNjZXNzQ29udHJvbEZhY3RvcnkuamF | `71.42% <ø> (-28.58%)` | :arrow_down: | | [.../helix/BrokerUserDefinedMessageHandlerFactory.java](https://codecov.io/gh/apache/incubator-pinot/pull/6425/diff?src=pr&el=tree#diff-cGlub3QtYnJva2VyL3NyYy9tYWluL2phdmEvb3JnL2FwYWNoZS9waW5vdC9icm9rZXIvYnJva2VyL2hlbGl4L0Jyb2tlclVzZXJEZWZpbmVkTWVzc2FnZUhhb | `33.96% <0.00%> (-32.71%)` | :arrow_down: | | [...ker/routing/instanceselector/InstanceSelector.java](https://codecov.io/gh/apache/incubator-pinot/pull/6425/diff?src=pr&el=tree#diff-cGlub3QtYnJva2VyL3NyYy9tYWluL2phdmEvb3JnL2FwYWNoZS9waW5vdC9icm9rZXIvcm91dGluZy9pbnN0YW5jZXNlbGVjdG9yL0luc3RhbmNlU2VsZWN | `100.00% <ø> (ø)` | | | [...ava/org/apache/pinot/client/AbstractResultSet.java](https://codecov.io/gh/apache/incubator-pinot/pull/6425/diff?src=pr&el=tree#diff-cGlub3QtY2xpZW50cy9waW5vdC1qYXZhLWNsaWVudC9zcmMvbWFpbi9qYXZhL29yZy9hcGFjaGUvcGlub3QvY2xpZW50L0Fic3RyYWN0UmVzdWx0U2V | `66.66% <0.00%> (+9.52%)` | :arrow_up: | | [.../main/java/org/apache/pinot/client/Connection.java](https://codecov.io/gh/apache/incubator-pinot/pull/6425/diff?src=pr&el=tree#diff-cGlub3QtY2xpZW50cy9waW5vdC1qYXZhLWNsaWVudC9zcmMvbWFpbi9qYXZhL29yZy9hcGFjaGUvcGlub3QvY2xpZW50L0Nvbm5lY3Rpb24uamF2YQ== | `35.55% <0.00%> (-13.29%)` | :arrow_down: | | [...inot/client/JsonAsyncHttpPinotClientTransport.java](https://codecov.io/gh/apache/incubator-pinot/pull/6425/diff?src=pr&el=tree#diff-cGlub3QtY2xpZW50cy9waW5vdC1qYXZhLWNsaWVudC9zcmMvbWFpbi9qYXZhL29yZy9hcGFjaGUvcGlub3QvY2xpZW50L0pzb25Bc3luY0h0dHBQaW5vd | `10.90% <0.00%> (-51.10%)` | :arrow_down: | | [...not/common/assignment/InstancePartitionsUtils.java](https://codecov.io/gh/apache/incubator-pinot/pull/6425/diff?src=pr&el=tree#diff-cGlub3QtY29tbW9uL3NyYy9tYWluL2phdmEvb3JnL2FwYWNoZS9waW5vdC9jb21tb24vYXNzaWdubWVudC9JbnN0YW5jZVBhcnRpdGlvbnNVdGlscy5qYX | `73.80% <ø> (+0.63%)` | :arrow_up: | | [...common/config/tuner/NoOpTableTableConfigTuner.java](https://codecov.io/gh/apache/incubator-pinot/pull/6425/diff?src=pr&el=tree#diff-cGlub3QtY29tbW9uL3NyYy9tYWluL2phdmEvb3JnL2FwYWNoZS9waW5vdC9jb21tb24vY29uZmlnL3R1bmVyL05vT3BUYWJsZVRhYmxlQ29uZmlnVHVuZ | `100.00% <ø> (ø)` | | | [...ot/common/config/tuner/RealTimeAutoIndexTuner.java](https://codecov.io/gh/apache/incubator-pinot/pull/6425/diff?src=pr&el=tree#diff-cGlub3QtY29tbW9uL3NyYy9tYWluL2phdmEvb3JnL2FwYWNoZS9waW5vdC9jb21tb24vY29uZmlnL3R1bmVyL1JlYWxUaW1lQXV0b1luZGV4VHVuZXIuam | `100.00% <ø> (ø)` | | | ... and [1154 more](https://codecov.io/gh/apache/incubator-pinot/pull/6425/diff?src=pr&el=tree-more) | | ------ [Continue to review full report at Codecov](https://codecov.io/gh/apache/incubator-pinot/pull/6425?src=pr&el=continue). > **Legend** - [Click here to learn more](https://docs.codecov.io/docs/codecov-delta) > `Δ = absolute <relative> (impact)`, `ø = not affected`, `? = missing data` > Powered by [Codecov](https://codecov.io/gh/apache/incubator-pinot/pull/6425?src=pr&el=footer). Last update [d04785c...ce187a8](https://codecov.io/gh/apache/incubator-pinot/pull/6425?src=pr&el=lastupdated). Read the [comment docs](https://docs.codecov.io/docs/pull-request-comments).

github_pulls_reviews:

jira_issues:

1. **summary:** Possible deadlock when creating new ClassInfo entries in the cache
   description: When running Groovy without {{-Dgroovy.use.classvalue=true}} the ClassInfo instances are cached in a {{ManagedConcurrentMap}} (MCM). New values are computed on demand and computation involves both a lock on a segment within the MCM and a lock on the {{GlobalClassSet}} (GCS) which is backed by a {{ManagedLinkedList}}. The problem is that both the ManagedConcurrentMap and the GlobalClassSet share the same ReferenceQueue. Assume there is an enqueued {{ClassInfo}} value that is stored in Segment2 of the MCM. Now assume that Thread1 and Thread2 both request {{ClassInfo.getClassInfo(..)}} for two different classes that do not currently exist in the cache. Assume that based on hashing Thread1 gets a lock on Segment1 and Thread2 gets a lock on Segment2. Assume that Thread1 is the first to call computeValue which in turn calls {{GlobalClassSet.add(..)}}. This call adds a new value to a {{ManagedLinkedList}}, and since it's managed the add operation will process the ReferenceQueue. So Thread1 will attempt to dequeue the ClassInfo and the finalizeReference method on it's entry will attempt to remove it from Segment2. Thread2 holds the lock for Segment2 and Thread2 is blocked and can't progress it's waiting on the the lock Thread1 holds the lock for the GlobalClassSet, so deadlock occurs. The attached test case includes a thread dump at the bottom.

jira_issues_comments:

1. This should only affect Groovy 2.4.8 and a possible workaround if using Java 7+ would be to run with {{-Dgroovy.use.classvalue=true}}.
2. GitHub user jwagenleitner opened a pull request: https://github.com/apache/groovy/pull/484 GROOVY-8067: Possible deadlock when creating new ClassInfo entries in the cache While I have been able to replicate the deadlock between `GroovyClassValuePreJava7$Segment` and the `GlobalClassSet#add`, I have not directly observed one between the `modifiedExpandos` and the `GlobalClassSet`, but think that it would be good to isolate their reference processing too since both lock in their operations. You can merge this pull request into a Git repository by running: $ git pull https://github.com/jwagenleitner/groovy groovy8067 Alternatively you can review and apply these changes as the patch at: https://github.com/apache/groovy/pull/484.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 #484 ---- commit 78f5aa0b5977919ba05dcad9fe8a7ee496abf2e8 Author: John Wagenleitner <jwagenleitner@apache.org> Date: 2017-01-29T18:26:43Z GROOVY-8067: test for demo only (DO NOT COMMIT) commit 58a27e7b1c437d436636658a5de9537fda5560d6 Author: John Wagenleitner <jwagenleitner@apache.org> Date: 2017-01-29T19:34:55Z GROOVY-8067: Possible deadlock when creating new ClassInfo entries in the cache ----
3. GitHub user jwagenleitner opened a pull request: https://github.com/apache/groovy/pull/489 GROOVY-8067: Possible deadlock when creating new ClassInfo entries in the cache As suggested in PR #484 removed the locking on the `ManagedLinkedList` by creating a new `ManagedConcurrentLinkedQueue`. Also added a `stress` subproject for tests that employ many threads, need GC, or just in general try to break things and take a long time. These require a special property to be set in order to run, otherwise they will be skipped. I tried to work it out in the `performance` subproject, but that seems to be very specialized for the compiler tests. Open to suggestions on a better way to handle these types of tests. You can merge this pull request into a Git repository by running: $ git pull https://github.com/jwagenleitner/groovy groovy8067-mclq Alternatively you can review and apply these changes as the patch at: https://github.com/apache/groovy/pull/489.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 #489 ---- commit bb2464a919a3655f36707fa72fb30080c92a7288 Author: John Wagenleitner <jwagenleitner@apache.org> Date: 2017-02-05T06:13:26Z GROOVY-8067: Possible deadlock when creating new ClassInfo entries in the cache ----
4. Github user jwagenleitner commented on a diff in the pull request: https://github.com/apache/groovy/pull/489#discussion_r99482851 --- Diff: src/main/org/codehaus/groovy/reflection/ClassInfo.java --- @@ -186,30 +185,20 @@ public void setStrongMetaClass(MetaClass answer) { MetaClass strongRef = strongMetaClass; if (strongRef instanceof ExpandoMetaClass) { - ((ExpandoMetaClass)strongRef).inRegistry = false; - synchronized(modifiedExpandos){ - for (Iterator<ClassInfo> it = modifiedExpandos.iterator(); it.hasNext();) { - ClassInfo info = it.next(); - if(info == this){ - it.remove(); - } + ((ExpandoMetaClass)strongRef).inRegistry = false; + for (Iterator<ClassInfo> itr = modifiedExpandos.iterator(); itr.hasNext();) { + ClassInfo info = itr.next(); + if(info == this) { + itr.remove(); + } } - } strongMetaClass = answer; if (answer instanceof ExpandoMetaClass) { - ((ExpandoMetaClass)answer).inRegistry = true; - synchronized(modifiedExpandos){ - for (Iterator<ClassInfo> it = modifiedExpandos.iterator(); it.hasNext();) { - ClassInfo info = it.next(); - if(info == this) { - it.remove(); - } - } - modifiedExpandos.add(this); - } --- End diff -- Removed because it appeared to be a duplicate of the logic above. Only if the previous previous `strongMetaClass` value was an expando would it have been added to the map, so the iteration/removal above should have taken care to remove this `ClassInfo` from the `modifiedExpandos` and there's no reason to iterate again.
5. Github user jwagenleitner closed the pull request at: https://github.com/apache/groovy/pull/484
6. Github user jwagenleitner commented on a diff in the pull request: https://github.com/apache/groovy/pull/489#discussion_r100162935 --- Diff: subprojects/stress/src/test/java/org/codehaus/groovy/reflection/ClassInfoDeadlockStressTest.java --- @@ -0,0 +1,140 @@ +/* + * 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. + */ +package org.codehaus.groovy.reflection; + +import groovy.lang.GroovyClassLoader; + +import java.util.concurrent.CountDownLatch; +import java.util.concurrent.TimeUnit; +import java.util.concurrent.atomic.AtomicInteger; + +import org.apache.groovy.stress.util.GCUtils; +import org.junit.Test; +import static org.junit.Assert.*; + +/** + * Tests for deadlocks in the ClassInfo caching. + * + */ +public class ClassInfoDeadlockStressTest { + + private static final int DEADLOCK_TRIES = 8; + private static final int THREAD_COUNT = 8; + + private final CountDownLatch startLatch = new CountDownLatch(1); + private final CountDownLatch completeLatch = new CountDownLatch(THREAD_COUNT); + private final GroovyClassLoader gcl = new GroovyClassLoader(); + private final AtomicInteger counter = new AtomicInteger(); + + /** + * We first generate a large number of ClassInfo instances for classes + * that are no longer reachable. Then queue up threads to all request + * ClassInfo instances for new classes simultaneously to ensure that + * clearing the old references wont deadlock the creation of new + * instances. + * <p> + * GROOVY-8067 + */ + @Test + public void testDeadlock() throws Exception { + for (int i = 1; i <= DEADLOCK_TRIES; i++) { + System.out.println("Test Number: " + i); + generateGarbage(); + GCUtils.gc(); + attemptDeadlock(null); + } + } + + @Test + public void testRequestsForSameClassInfo() throws Exception { + Class<?> newClass = createRandomClass(); + for (int i = 1; i <= DEADLOCK_TRIES; i++) { + System.out.println("Test Number: " + i); + generateGarbage(); + GCUtils.gc(); + attemptDeadlock(newClass); + } + ClassInfo newClassInfo = ClassInfo.getClassInfo(newClass); + for (ClassInfo ci : ClassInfo.getAllClassInfo()) { + if (ci.getTheClass() == newClass && ci != newClassInfo) { + fail("Found multiple ClassInfo instances for class"); + } + } + } + + private void attemptDeadlock(final Class<?> cls) throws Exception { + for (int i = 0; i < THREAD_COUNT; i++) { + Runnable runnable = new Runnable() { + @Override + public void run() { + Class<?> newClass = (cls == null) ? createRandomClass() : cls; + try { + startLatch.await(); + } catch (InterruptedException ie) { --- End diff -- can use `ThreadUtils.awaite()`
7. Github user jwagenleitner commented on a diff in the pull request: https://github.com/apache/groovy/pull/489#discussion_r100162813 --- Diff: src/main/org/codehaus/groovy/util/ManagedConcurrentLinkedQueue.java --- @@ -0,0 +1,187 @@ +/* + * 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. + */ +package org.codehaus.groovy.util; + +import java.util.ArrayList; +import java.util.Collections; +import java.util.Iterator; +import java.util.List; +import java.util.NoSuchElementException; +import java.util.concurrent.ConcurrentLinkedQueue; + +/** + * A queue that stores the values wrapped in a Reference, the type of which is + * determined by the provided {@link ReferenceBundle}. Values stored in this queue + * that are put on the {@code ReferenceQueue} will be removed from the list when + * reference processing for the {@code ReferenceQueue} is done. + * + * This queue is backed by a {@link ConcurrentLinkedQueue} and is thread safe. The + * iterator will only return non-null values (reachable) and is based on the + * "weakly consistent" iterator of the underlying {@link ConcurrentLinkedQueue}. + * + * @param <T> the type of values to store + */ +public class ManagedConcurrentLinkedQueue<T> implements Iterable<T> { + + private final ReferenceBundle bundle; + private final ConcurrentLinkedQueue<Element<T>> queue; + + /** + * Creates an empty ManagedConcurrentLinkedQueue that will use the given + * {@code ReferenceBundle} to store values as the given Reference + * type. + * + * @param bundle used to create the appropriate Reference type + * for the values stored + */ + public ManagedConcurrentLinkedQueue(ReferenceBundle bundle) { + this.bundle = bundle; + this.queue = new ConcurrentLinkedQueue<Element<T>>(); + } + + /** + * Adds the specified value to the queue. + * + * @param value the value to add + */ + public void add(T value) { + Element<T> e = new Element<T>(value); + queue.offer(e); + } + + /** + * Returns {@code true} if this queue contains no elements. + * <p> + * This method does not check the elements to verify they contain + * non-null reference values. + */ + public boolean isEmpty() { + return queue.isEmpty(); + } + + /** + * Returns an array containing all values from this queue in the sequence they + * were added. + * + * @param tArray the array to populate if big enough, else a new array with + * the same runtime type + * @return an array containing all non-null values in this queue + */ + public T[] toArray(T[] tArray) { + return values().toArray(tArray); + } + + /** + * Returns an unmodifiable list containing all values from this queue in the + * sequence they were added. + */ + public List<T> values() { + Iterator<T> itr = iterator(); + if (!itr.hasNext()) { + return Collections.emptyList(); + } + List<T> result = new ArrayList<T>(100); + result.add(itr.next()); + while (itr.hasNext()) { + result.add(itr.next()); + } + return Collections.unmodifiableList(result); + } + + /** + * Returns an iterator over all non-null values in this queue. The values should be + * returned in the order they were added. + */ + @Override + public Iterator<T> iterator() { + return new Iter(queue.iterator()); + } + + private final class Element<V> extends ManagedReference<V> { + + Element(V value) { + super(bundle, value); + } + + @Override + public void finalizeReference() { + queue.remove(this); + super.finalizeReference(); + } + } + + private final class Iter implements Iterator<T> { --- End diff -- think this can be made a static nested class
8. Github user asfgit closed the pull request at: https://github.com/apache/groovy/pull/489
9. Git bisect points to the respective commit for this issue for the below Grails plugin issue: https://github.com/grails/grails-core/issues/10715 I haven't fully investigated and don't have a standalone reproducer yet. I'll raise a separate issue when I find something conclusive.
10. It looks like the issue might be related to the {{ManagedLinkedList}} (MLL) iterator vs the {{ManagedConcurrentLinkedQueue}} (MCLQ) iterator. The MLL {{iterator#remove}} method seems to not correctly relink the list (head, tail). {code} import org.codehaus.groovy.util.* mml = new ManagedLinkedList<String>(ReferenceBundle.getHardBundle()) //mml = new ManagedConcurrentLinkedQueue<String>(ReferenceBundle.getHardBundle()) mml.add('foo') mml.add('bar') mml.add('baz') for (Iterator<String> itr = mml.iterator(); itr.hasNext();) { String s = itr.next() println s itr.remove() } {code} Output from MLL: {code} foo {code} Output from MCLQ: {code} foo bar baz {code} So with the old code using MLL any calls to {{clearModifiedExpandos}} would always only remove the first one.

The [call to ExpandoMetaClass.enableGlobally in the grails-melody-plugin|https://github.com/javamelody/grails-melody-plugin/blob/4da5c1e7092f7841e6133fd790baa0419340a6ad/src/main/groovy/grails/melody/plugin/MelodyInterceptorEnhancer.groovy#L32] triggers a call to {{clearModifiedExpandos}}. By this time Grails core has already added the {{encode}} methods. The old code just by luck would not remove the modified expandos (and more importantly is the call to {{setStrongMetaClass(null)}}) but the new MCLQ really does clear all modified expandos. Assuming that's the problem, not sure what's the appropriate fix. It seems to me like the old MLL is broken, it doesn't look like this would have been the intended behavior. So guess the question would be whether the responsibility lies with the caller of {{ExpandoMetaClass#enableGlobally}} to check first to see if already enabled or whether that method should not clear if already enabled.

11. {quote} So guess the question would be whether the responsibility lies with the caller of {{ExpandoMetaClass#enableGlobally}} to check first to see if already enabled or whether that method should not clear if already enabled. {quote} The enableGlobally call does already check and only clears if the {{ExpandoMetaClassCreationHandle}} is not the current handle. In tracing the calls to clearModifiedExpandos when running the sample project grails-javamelody-issue the plugin triggers 2 calls to clearModifiedExpandos, first call in {{ExpandoMetaClassCreationHandle#create}} (via {{ExpandoMetaClass.enableGlobally}}) and second one in {{MetaClassRegistryImpl#setMetaClassCreationHandle}} (via the before mentioned create call). So even with the old {{ManagedLinkedList}} (in Groovy 2.4.8) the plugin would have removed 2 modified expandos, those for {{org.grails.plugins.codecs.URLCodec}} and {{org.grails.encoder.impl.JavaScriptCodec}}.