Infrastructure - Bug #8031

when synchronization reaches max lock attempts, it keeps going

2017-03-01 20:51 - Rob Nahf

Status:	Closed	**Start date:**	2017-03-01
Priority:	Normal	**Due date:**	
Assignee:	Rob Nahf	**% Done:**	100%
Category:	d1_synchronization	**Estimated time:**	0.00 hour
Target version:	CCI-2.3.2		
Milestone:	None	**Story Points:**	
Product Version:	*		

Description

For one pid, the task just kept being retried endlessly with this logged message:

[WARN] 2017-02-27 09:20:39,505 (SyncFailedTask:submitSynchronizationFailed:116)
Task-urn:node:ARCTIC-urn:uuid:ed2f771b-6c29-45c1-8f46-24df0b35d0db <?xml version="1.0" encod
ing="UTF-8"?>

Cannot lock Pid! Reached Max attempts (100), abandoning processing of this pid.

[DEBUG] 2017-02-27 09:20:39,550 (SyncObjectTask:reapFutures:334)
Task-urn:node:ARCTIC-urn:uuid:ed2f771b-6c29-45c1-8f46-24df0b35d0db Waiting for the future :(2): since 2017-0
2-27T09:20:37.544+00:00

Associated revisions

Revision 18646 - 2017-03-01 20:55 - Rob Nahf

fixes #8031: endless processing of failing pid.

Revision 18646 - 2017-03-01 20:55 - Rob Nahf

fixes #8031: endless processing of failing pid.

Revision 18647 - 2017-03-03 19:51 - Rob Nahf

fixes #8031: fixes thread-safety issue with lock acquisition and unlocking.

Revision 18647 - 2017-03-03 19:51 - Rob Nahf

fixes #8031: fixes thread-safety issue with lock acquisition and unlocking.

Revision 19031 - 2017-11-17 18:59 - Rob Nahf

fixes #8031: found the bug in SyncObjectTask that was causing infinite synchronization loop after de-activating sync. Minor adjustment on polling timeouts. Fixed minor seriesId validating bug.

Revision 19031 - 2017-11-17 18:59 - Rob Nahf

fixes #8031: found the bug in SyncObjectTask that was causing infinite synchronization loop after de-activating sync. Minor adjustment on polling timeouts. Fixed minor seriesId validating bug.

History

#1 - 2017-03-01 20:53 - Rob Nahf

- Status changed from In Progress to Testing

- % Done changed from 30 to 50

There is a bug in the code that is checking the wrong counter, so never breaks out of the loop.

(I thought I fixed this a while ago - maybe the change was overwritten?)

#2 - 2017-03-01 22:04 - Rob Nahf

- Status changed from Testing to Closed

- % Done changed from 50 to 100

Applied in changeset java-client:d1client|r18646.

#3 - 2017-03-02 00:22 - Rob Nahf

Comment #1 is incorrect - there was a committed fix in branch 2.3 from Dec 2016.

So, the code looks pretty solid, except that SyncObjectTask (which calls V2TransferObjectTask) is run as a daemon thread, and daemon threads are abandoned when the JVM exits, and can cause finally blocks to be skipped. (see http://stackoverflow.com/a/2213443) In this case, the finally block unlocks the object. Because it is a distributed lock (Hazelcast) restarting sync rejoins Hazelcast where the lock is set.

This would explain why the finally block is not run, but not why the task was run thousands of times, presumably within the same execution, judging by the log files.

#4 - 2017-03-02 23:28 - Rob Nahf

- % Done changed from 100 to 30

- Status changed from Closed to In Progress

#5 - 2017-03-03 19:59 - Rob Nahf

- % Done changed from 30 to 50

- Status changed from In Progress to Testing

there's a problem with the lock setting and unlocking try-catch-finally structure that looks like it was allowing a thread to set a lock and not unlock it. see second example on http://docs.hazelcast.org/docs/3.5/manual/html/lock.html.

(use of the boolean isLockAcquired caused a disassociation of the lock acquisition with the conditional.)

hopefully fixed.

#6 - 2017-03-03 20:12 - Rob Nahf

- Status changed from Testing to Closed

- % Done changed from 50 to 100

Applied in changeset d1-infrastructure|r18647.