[2.2.0] org.neo4j.kernel.impl.store.InvalidRecordException: DynamicRecord Not in use

New issue

Open **tkroman** opened this issue Apr 8, 2015 · 10 comments

Labels
None yet

Milestone
No milestone

Assignee
No one assigned

3 participants

tkroman commented Apr 8, 2015

We encountered this bug during a long-running test. I grep'd messages.log on the subject, but there's nothing pertaining to the issue. Please let me know if I could provide you with anything else.

```
java.lang.RuntimeException: org.neo4j.kernel.impl.store.InvalidRecordException: DynamicRec
at org.neo4j.kernel.api.properties.LazyProperty.produceValue(LazyProperty.java:73)
at org.neo4j.kernel.api.properties.LazyProperty.value(LazyProperty.java:58)
at org.neo4j.kernel.impl.core.NodeProxy.getProperty(NodeProxy.java:396)
at dnet.controller.services.files.Allocate.rpc(Allocate.java:58)
....
Caused by: org.neo4j.kernel.impl.store.InvalidRecordException: DynamicRecord Not in use, b
at org.neo4j.kernel.impl.store.AbstractDynamicStore.checkForInUse(AbstractDynamicStore.jav
at org.neo4j.kernel.impl.store.AbstractDynamicStore.getRecords(AbstractDynamicStore.java:4
at org.neo4j.kernel.impl.store.AbstractDynamicStore.getLightRecords(AbstractDynamicStore.j
at org.neo4j.kernel.impl.store.PropertyStore.ensureHeavy(PropertyStore.java:295)
at org.neo4j.kernel.impl.store.PropertyStore.getStringFor(PropertyStore.java:601)
at org.neo4j.kernel.impl.store.PropertyType$9.getValue(PropertyType.java:213)
at org.neo4j.kernel.impl.store.PropertyType$9$1.call(PropertyType.java:201)
at org.neo4j.kernel.impl.store.PropertyType$9$1.call(PropertyType.java:197)
at org.neo4j.kernel.api.properties.LazyProperty.produceValue(LazyProperty.java:69)
... 14 more
```

jexp commented Apr 8, 2015 Collaborator

Can you share more details about you graph structure, workload, if the db was migrated, if this also happens if you just scan the db, consistency checker output , etc
http://www.markhneedham.com/blog/2014/01/22/neo4j-backup-store-copy-and-consistency-check/

tkroman commented Apr 8, 2015

The database has a tree-like structure. It's tree-*like* because there are other types of relationships that form edges of the tree. Actually, this is irrelevant because we had 2 exceptions of this kind over the course of the night test and both times the nodes causing them weren't related to any other nodes (we use them to temporarily store data that will be persisted later, after some BL checks succeed). These nodes have their own label (say `Util`), and there's a unique constraint for `:Util:uuid`. These nodes also have the property `path`, which is later used to insert node into the specific place in the tree. Both times the exception happened during the call

```
Object path = node.getProperty(File.PATH_ATTR);
```

The amount of nodes created was some 2 to 3 million nodes in the trees, hence there were another 1-1.8M short-lived `Util` nodes. (The TTL of such a node is rarely over 5 seconds).

The database wasn't migrated.

We also fail fast on the unexpected exceptions in the place where this happened to retry later, and I have enough information to assert that the subsequent retry succeeded.

This happened at around 6 AM, and the night test continued for another 4 to 6 hours - with corresponding database usage etc., so I can also assert that this issue didn't cause some kind of a total DB files corruption.

I'll give consistency check tool a try and let you know, but IMO the database is intact.

jexp commented Apr 8, 2015 Collaborator

Do you use it remotely or embedded? And how do you control your transactions? Could be a kind of race condition

tkroman commented Apr 8, 2015

We use it embedded into the main application. Transactions are controlled manually. We have 2 services that may access these nodes concurrently. Both are ~20 lines long and perform the following - one reads nodes' properties and fires another event and another gets nodes and deletes them. These services open their corresponding transactions, which are subsequently closed as usually. We also catch and log `NotFounExceptions` to track errors, but nothing showed up around that time.

BTW, messages.log contains this:

```
2015-04-08 03:16:16.981+0000 WARN  [o.n.k.h.HighlyAvailableGraphDatabase]: GC Monitor: App
2015-04-08 03:16:24.976+0000 INFO  [o.n.k.i.a.i.s.OnlineIndexSamplingJob]: Sampled index :
2015-04-08 03:16:26.962+0000 WARN  [o.n.k.h.HighlyAvailableGraphDatabase]: GC Monitor: App
2015-04-08 03:16:40.579+0000 WARN  [o.n.k.h.HighlyAvailableGraphDatabase]: GC Monitor: App
2015-04-08 03:16:41.535+0000 INFO  [o.n.k.i.c.Caches]: NodeCache purge (nr 1348) 284.5466M
2015-04-08 03:16:42.258+0000 INFO  [o.n.k.i.c.Caches]: NodeCache purge (nr 1348): elementC
2015-04-08 03:16:53.542+0000 WARN  [o.n.k.h.HighlyAvailableGraphDatabase]: GC Monitor: App
```

and exceptions were thrown between:

1. `2015-04-08 03:16:20,290` and `2015-04-08 03:16:20,414`
2. `2015-04-08 03:16:20,490` and `2015-04-08 03:16:20,513`

In case this might help.

tinwelint commented Apr 13, 2015 Collaborator

Was this exception temporary or did it get thrown over and over? There are scenarios like these that might occur since neo4j doesn't acquire read locks when reading. It looks like the exception might come from two threads `A` and `B` racing. `A` wants to read some property and since that property hasn't been accessed on that cached node, it will go and load that property from store. At the exact same moment `B` is in the commit stage of deleting this property. Before `B` updates the cache `A` has already failed trying to load it from the store.

This is definitely something that should be handled better.

tkroman commented Apr 13, 2015

It happened once. Your scenario looks very similar to ours except the fact that we delete not a single property but the node itself. I tried to reproduce this by carefully timing concurrent read/write operations but unfortunately I couldn't.
I'd be really glad to tell you more, so if there's some way to learn additional information, I'll set up a try-catch block at a minimal scope possible to try and log any information that might be useful should this happen again.

tinwelint commented Apr 13, 2015 Collaborator

Deleting the node would effectively produce the same scenario. It should be fairly easy to create a test that produces this behaviour in a non-tererministic way by having a node with a property `P` with a big string or a `String[]` with at least a couple of elements and having called `getProperty` on *some other* property. Then coordinating two threads at the same time: one that commits the deletion of the node and the other that does `getProperty` on `P`.

tkroman commented Apr 13, 2015

Well then, I'll give it a try as soon as I can.

tkroman commented Apr 15, 2015

Yep, I got it:

```
Exception in thread "pool-4-thread-1" java.lang.RuntimeException: org.neo4j.kernel.impl.st
at org.neo4j.kernel.api.properties.LazyProperty.produceValue(LazyProperty.java:73)
at org.neo4j.kernel.api.properties.LazyArrayProperty.produceValue(LazyArrayProperty.java:5
at org.neo4j.kernel.api.properties.LazyProperty.value(LazyProperty.java:58)
at org.neo4j.kernel.impl.core.NodeProxy.getProperty(NodeProxy.java:396)
at ontrack.bush.BushTreeTest.lambda$testDynamicRecordIssues$2(BushTreeTest.java:345)
at ontrack.bush.BushTreeTest$$Lambda$25/2118904117.run(Unknown Source)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
Caused by: org.neo4j.kernel.impl.store.InvalidRecordException: DynamicRecord Not in use, b
at org.neo4j.kernel.impl.store.AbstractDynamicStore.checkForInUse(AbstractDynamicStore.jav
at org.neo4j.kernel.impl.store.AbstractDynamicStore.getRecords(AbstractDynamicStore.java:4
at org.neo4j.kernel.impl.store.AbstractDynamicStore.getLightRecords(AbstractDynamicStore.j
at org.neo4j.kernel.impl.store.PropertyStore.ensureHeavy(PropertyStore.java:311)
at org.neo4j.kernel.impl.store.PropertyStore.getArrayFor(PropertyStore.java:614)
at org.neo4j.kernel.impl.store.PropertyType$10.getValue(PropertyType.java:244)
at org.neo4j.kernel.impl.store.PropertyType$10$1.call(PropertyType.java:232)
at org.neo4j.kernel.api.properties.LazyProperty.produceValue(LazyProperty.java:69)
... 8 more
```

The test is non-deterministic, as you said: I managed to get the exception for just a 40% of launches (the rest throw `NotFoundException`), but here it is:

```
@Test
public void testDynamicRecordIssues() throws Exception {
    Label l = () -> "testDynamicRecordIssues";
    ExecutorService pool = Executors.newFixedThreadPool(2);
    Node node;
    try (final Transaction tx = db.beginTx()) {
        node = db.createNode(l);
        node.setProperty("p", new String[]{"qwertyuiop", "asdfghjkl", "zxcvbnm", "qazwsx",
        node.setProperty("k", "v");
        node.setProperty("x", "y");
        tx.success();
    }

    Semaphore s = new Semaphore(0);

    pool.execute(() -> {
        try (final Transaction tx = db.beginTx()) {
            node.getProperty("x");
            s.acquire();
            TimeUnit.MILLISECONDS.sleep(2);
            System.out.println(Arrays.toString((String[]) node.getProperty("p")));
            tx.success();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    });

    pool.execute(() -> {
        Transaction tx = db.beginTx();
        try {
            node.delete();
            tx.success();
        } finally {
            s.release();
            tx.close();
        }
    });

    pool.shutdown();
    pool.awaitTermination(5, TimeUnit.SECONDS);
}
```

The main point of pressure during the reproduction should be the `TimeUnit.MILLISECONDS.sleep(2);`

line: I had to justify it a couple of times and I'm not sure that it will fit your hardware performance or whatever.

tinwelint commented Apr 20, 2015 Collaborator

Awesome, a good baseline test to start working from. With this, hopefully this problem will be made nicer soon.

arikan referenced this issue Jul 29, 2015

InvalidRecordException: DynamicRecord Not in use #5055 Open

Sign up for free **to join this conversation on GitHub**. Already have an account? Sign in to comment