Alfresco repo under concurrent write load

Alexey Vasyukov, ITD Systems

The System

- ✓ Alfresco repo under reasonable write load
- ✓ In our experience we have the following:
  - ✓ Alfresco 5.2
  - ✓ 20 000 users
  - ✓ 10-100 write transactions per second

Pseudo-code of simple write-intensive API

```
// … classify content, determine target location …
checkForExistingFolder();
if(!folderExists) {
    createFolder();
} else {
    useExistingFolder();
}
// … do real stuff with incoming content …
```

Good code, works on my machine!

During basic load testing

```
python load.py --api demo1 --concurrency 5
demo1: 200
demo1: 500
demo1: 500
demo1: 500
demo1: 500
```

```
org.alfresco.service.cmr.repository.DuplicateChildNodeNameException:
Duplicate child name not allowed: WRITE_CONCURRENCY_DEMO_1
```

```
checkForExistingFolder();
if(!folderExists) {
    createFolder();
} else {
    useExistingFolder();
}
```

Got it! Easy!

Concurrency issue!

```
2017-04-22 18:10:44,200  INFO Started new request processing
2017-04-22 18:10:44,201  INFO Started new request processing
2017-04-22 18:10:44,206  INFO Started new request processing
2017-04-22 18:10:44,206  INFO Started new request processing
2017-04-22 18:10:44,210  INFO Started new request processing
2017-04-22 18:10:44,304  INFO Creating new folder
2017-04-22 18:10:44,308  INFO Creating new folder
2017-04-22 18:10:44,310  INFO Creating new folder
2017-04-22 18:10:44,311  INFO Creating new folder
2017-04-22 18:10:44,317  INFO Creating new folder
```

Concurrent writes

So, we need to synchronize... No, you do not!

"Alfresco uses "optimistic" locking so the approach is attempt to update and if there is conflict, fail fast and retry."

https://community.alfresco.com/thread/180405-behaviour-threading-doesnt-make-sense

Official option: RetryingTransactionHelper

Docs:	http://docs.alfresco.com/5.2/references/dev-extension-points-public-java-api.html
Discussion:	https://community.alfresco.com/thread/153001-how-to-write-code-with-transactions

RetryingTransactionHelper Official Way

```
serviceRegistry.getRetryingTransactionHelper().doInTransaction(
  new RetryingTransactionHelper.RetryingTransactionCallback<NodeRef>() {
    public NodeRef execute() throws Throwable {
        // … classify content, determine target location …
        checkForExistingFolder();
        if(!folderExists) {
            createFolder();
        } else {
            useExistingFolder();
        }
        // … do real stuff with incoming content …
    }
  }, false, true);
```

Ok, let's test it

```
python load.py --api demo2 --concurrency 5
demo1: 200
demo1: 500
demo1: 500
demo1: 500
demo1: 500
```

```
org.alfresco.service.cmr.repository.DuplicateChildNodeNameException:
Duplicate child name not allowed: WRITE_CONCURRENCY_DEMO_2
```

Wait... But...

```
org.alfresco.service.cmr.repository.DuplicateChildNodeNameException:
Duplicate child name not allowed: WRITE_CONCURRENCY_DEMO_2
...
at RetryingCallbackHelper.doWithRetry(RetryingCallbackHelper.java:101)
...
at RetryingTransactionHelper.doInTransaction(RetryingTransactionHelper
.java:333)
at WriteConcurrencyDemo2.executeImpl(WriteConcurrencyDemo2.java:44)
...
```

Strange. Ok. I go with old good approach.

```
serviceRegistry.getRetryingTransactionHelper().doInTransaction(
  new RetryingTransactionHelper.RetryingTransactionCallback<NodeRef>() {
    public NodeRef execute() throws Throwable {
        // … classify content, determine target location …
        checkForExistingFolder();
        if(!folderExists) {
            try {
                createFolder();
            } catch (DuplicateChildNodeNameException e) {
                useExistingFolder();
            }
        } else {
            useExistingFolder();
        }
        // … do real stuff with incoming content …
    }
  }, false, true);
.
```

Not nice.

But should be stable.

Test this

```
python load.py --api demo3 --concurrency 5
demo1: 200
demo1: 200
demo1: 200
demo1: 200
demo1: 200
```

Ok. Works.

Not nice code, but works as expected.

Accidentally look in the repo

```
/app:company_home/cm:WRITE_CONCURRENCY_DEMO_3


Children (1):
cm:df1eb6b8-0304-4db9-9e2f-0eeac77e007d …
```

Oh. Wait!

There should be 5 elements! 5!

Long story short

Any exception (even caught one!) marks transaction for rollback:

http://stackoverflow.com/questions/19302196/transaction-marked-as-rollback-only-how-do-i-find-the-cause

Repo rolls transaction back and considers this normal operation:

https://github.com/Alfresco/community-edition/blob/master/projects/repository/source/java/org/alfresco/repo/transaction/RetryingTransactionHelper.java#L480

You catch and handle exception.

It rollbacks transaction. Considers it normal operation.

Throws your data. Silently. Code 200. No error in log.

Keep calm and go deeper

There is an option (not documented, not even mentioned):

```
<property name="extraExceptions">
  <list>
    <value>
      org.alfresco.service.cmr.repository.DuplicateChildNodeNameException
    </value>
    <value>
      org.alfresco.service.cmr.repository.InvalidNodeRefException
    </value>
    . . . whatever . . .
  </list>
</property>
```

Towards happy end

You can not get correct bean from transactionService. Create it manually.

```
<bean id="demo4.retryingTransactionHelper"
      class="org.alfresco.repo.transaction.RetryingTransactionHelper">
  <property name="extraExceptions">
    . . . whatever, your list of custom exceptions . . .
  </property>
  . . . other properties, if necessary . . .
</bean>
```

Towards happy end

```
retryingTransactionHelper.doInTransaction(
  new RetryingTransactionHelper.RetryingTransactionCallback<NodeRef>() {
    public NodeRef execute() throws Throwable {
        // … classify content, determine target location …
        checkForExistingFolder();
        if(!folderExists) {
            createFolder();
        } else {
            useExistingFolder();
        }
        // … do real stuff with incoming content …
    }
  }, false, true);
```

Injected custom bean

Crucial params

Test again

```
python load.py --api demo4 --concurrency 5
demo1: 200
demo1: 200
demo1: 200
demo1: 200
demo1: 200
```

Hell yeah. Finally.

```
/app:company_home/cm:WRITE_CONCURRENCY_DEMO_4


Children (5):
cm:df1eb6b8-0304-4db9-9e2f-0eeac77e007d …
. . .
```

Closing remarks

- ✓ Examples provided are based on simple “create folder” problem
- ✓ The same problem exists for real life cases (any exception!)
- ✓ Not a bug, optimistic locking feature
- ✓ Sources to play live: https://github.com/avasyukov/beecon2017-write-concurrency-demo

Alfresco Repo can be stable under concurrent write load.

Be really careful with exception *(consider third-party code)*.

Alexey Vasyukov

avasyukov@itdhq.com