May 28 2012

Simulating Rollback on MongoDB

Techno Geek Add comments

The Rollback state of MongoDB can cause a lot of confusion when it occurs. While there are documents and posts that describe how to deal with the state, there is nothing that explains step by step how you might end up that way. I often find that figuring out how to simulate the event makes it a lot clearer what is going on, while at the same time allowing people to figure out how they should deal with the situation when it arises. Here is the basic outline:

1. Create a replica set, load up some data, verify it is working as expected, start to insert data
2. Break replication (artificially) on the set so that the secondary falls behind
3. Stop the primary while the secondary is still lagging behind
4. This will cause the lagging secondary to be promoted

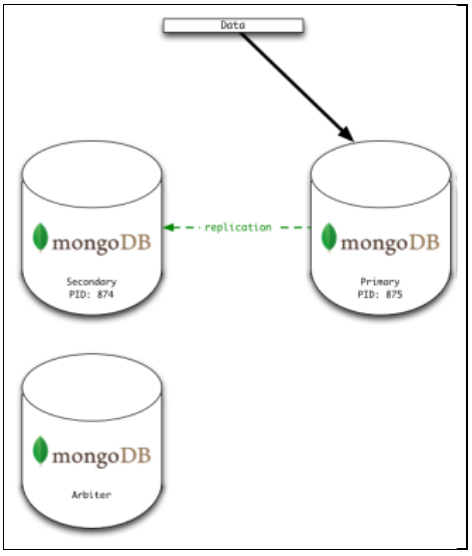
I'll go through each step, including diagrams to explain what is going on after the jump.

———

First, lets take a very simple replica set – Primary/Secondary/Arbiter. Personally I run the whole set on a single machine for my testing purposes using the mongo-snippets replication simple-setup.py script. My command looks like this:

```
simple-setup.py --mongo_path=/Users/adam/mongo/2.0.5/bin/ --arbiters=1
```

That gives us an initial set up something like this:



The first thing we need to do is start inserting some data so that it will be replicated. We don't need a lot of data to make this work, so just use a simple for loop in the mongo shell to insert 10 records a second:

```
for(var i = 0; i <= 1000000 ; i++){db.rollback.insert({"a" : i}); sleep(100);}
```

Next we need to make the secondary lag while data continues to be inserted to the primary. This is actually quite easily done, we just send a SIGSTOP to the secondary mongod process (in the diagram this is "PID 874". So the command would look like this:

```
kill -SIGSTOP 874
```

That effectively breaks replication but leaves the secondary "UP" and easily restarted without having to

Social Media

@comerford on Twitter

Pages

About Adam Comerford
Fixing Gigabyte GA-X58A-UD3R Endless Reboot Issue
Home Printing – Connected to a VPN
LG CU500 Hacking
Online Resume/CV
Ubuntu on Sun Blade 100/150
UJDA710 Flashing Guide

Adam on Flickr




Blog Linkage

Ait Eigin
Dave Bushe
Grateful Dating
Illyana
Just Joe
LeSinge
Louis
Patrick O'Leary
Ryan
Shane Donnelly
The Kid From Crumlin
Tormentron!

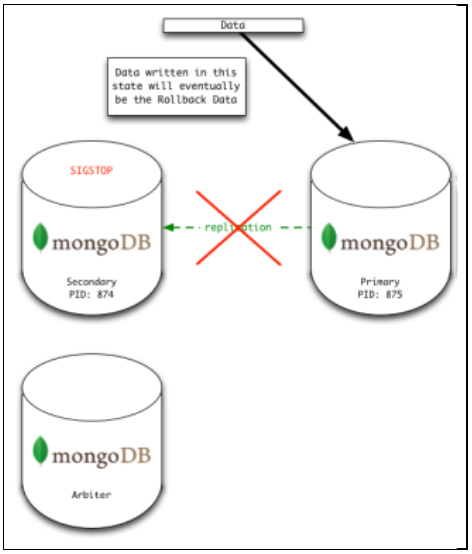
More of Me

Adam on Facebook
Adam on Flickr
Adam on Yelp
Adam's MySpace
Linked In
Work Blog

Older Content

Blogger to RSS
Julia's Site
Urban ISP Recovery

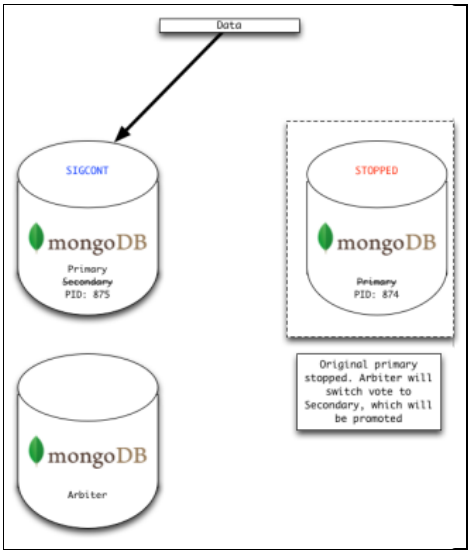
manually re-invoke it. Our replica set looks now looks like this:



As mentioned in the diagram, the data written while in this state will eventually be our rollback data. The longer we leave the set in this state, the bigger that data set will become. To make the rollback happen, we now need to stop the current primary and have the incoming data flip over to the secondary. Again we will achieve this with a signal, but this time it will be a SIGINT, the equivalent of pressing Ctrl-C on the command line, while at the same time immediately sending a SIGCONT to bring the secondary back to normal:

```
kill -SIGINT 875; kill -SIGCONT 874
```

This will interrupt your for loop in the shell, because you have killed the primary you were using to do the inserts. In a driver or an app, this would switch over automatically when the promotion happens, but we need to intervene and connect to the new primary, then restart our loop to keep inserts going. Once you have done so, the set will look like this:



Finally, we need to restart the original primary and bring it back into the set. To do so, just find the original options for this set – if you started as I have done, these were printed on your console as the simple-setup.py script started it up. Look for something like this for the appropriate instance (there will be one for each):

```
R1: Mon May 28 13:12:15 [initandlisten] options: { dbpath: "/data/db/replset/rs_2", c
```

This translates to:

/path/to/mongod –dbpath “/data/db/replset/rs_2″ –oplogSize 100 –port 27019 –replSet “foo/localhost:27017,localhost:27018″ –rest

Once the instance starts up, the set will (briefly) look like this:

Pictures

- Clydes 2004
- Picture Gallery
- Portugal 2003
- The Car
- Twins 21st

Web Sites

- Penny Arcade
- Swanny-isms

Archives

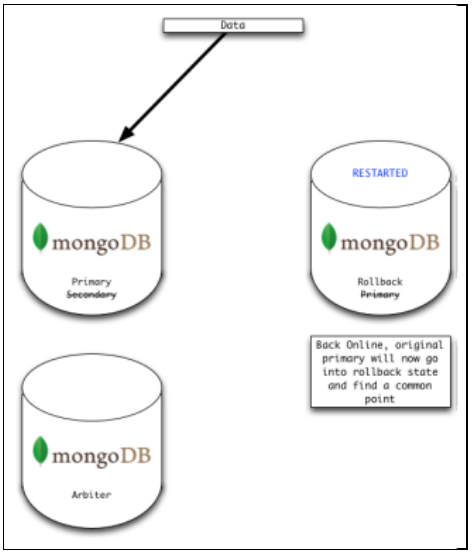
Select Month

Tags from Posts

.cc Arlington baby bapper baptacular bmw brian Caitlinn Capitol Hill charity CLIENT-XFER-PROHIBITED dcist domain transfer Dr. Who Eddie Izzard elections enom gamification godaddy Hollywood Video iphone Ireland Labour mobile phone mongodb MongoDB, Inc. Netscaler O2 Ood Paddy Tax Pictures pictures "New Orleans" politics rant Redskins Reed registrar Rip Off Ireland road trip shopping electronics drobo leaving sleeping stereophonics twitter Washington DC Year of Giving

Admin Stuff

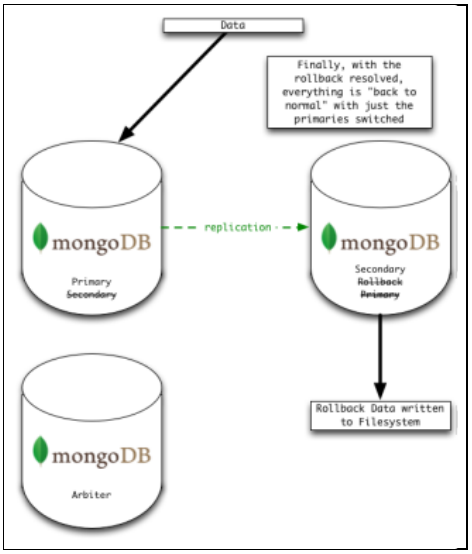
- Log in
- Entries RSS
- Comments RSS
- WordPress.org



Finally, you will see messaging for your restarted instance indicating the rollback has occurred, a common point has been found and the data has been reconciled. As an example, here is my output:

```
Mon May 28 13:46:19 [rsSync] replSet syncing to: localhost:27018
Mon May 28 13:46:19 [rsSync] replSet our last op time written: May 28 13:32:11:5
Mon May 28 13:46:19 [rsSync] replset source's GTE: May 28 13:32:43:1
Mon May 28 13:46:19 [rsSync] replSet rollback 0
Mon May 28 13:46:19 [rsSync] replSet ROLLBACK
Mon May 28 13:46:19 [rsSync] replSet rollback 1
Mon May 28 13:46:19 [rsSync] replSet rollback 2 FindCommonPoint
Mon May 28 13:46:19 [rsSync] replSet info rollback our last optime: May 28 13:32:11:5
Mon May 28 13:46:19 [rsSync] replSet info rollback their last optime: May 28 13:46:19
Mon May 28 13:46:19 [rsSync] replSet info rollback diff in end of log times: -848 sec
Mon May 28 13:46:19 [rsSync] replSet rollback found matching events at May 28 13:20:5
Mon May 28 13:46:19 [rsSync] replSet rollback findcommonpoint scanned : 14743
Mon May 28 13:46:19 [rsSync] replSet replSet rollback 3 fixup
Mon May 28 13:46:19 [rsSync] replSet rollback 3.5
Mon May 28 13:46:19 [rsSync] replSet rollback 4 n:6666
Mon May 28 13:46:19 [rsSync] replSet rollback 4.6
Mon May 28 13:46:19 [rsSync] replSet rollback 4.7
Mon May 28 13:46:20 [rsSync] replSet rollback 5 d:13332 u:0
Mon May 28 13:46:20 [rsSync] replSet rollback 6
Mon May 28 13:46:20 [rsSync] replSet rollback 7
Mon May 28 13:46:20 [rsSync] replSet rollback done
Mon May 28 13:46:20 [rsSync] replSet RECOVERING
Mon May 28 13:46:21 [rsSync] replSet syncing to: localhost:27018
Mon May 28 13:46:21 [rsSync] replSet still syncing, not yet to minValid optime4fc3739
Mon May 28 13:46:21 [rsSync] replSet SECONDARY
```

Once that is complete we end up with a set something like what we started out with, but with the Primaries flipped:



And that's it, the rollback data is available for you to inspect/restore in the rollback folder, which looked like this for me:

```
$ pwd
/data/db/replset/rs_2/rollback
$ ls -lh
total 432
-rw-r--r-- 1 adam staff 215K 28 May 13:46 test.rollback.2012-05-28T12-46-19.0.bson
```

That data can now be re-inserted, restored to a new collection etc. – the set will keep on working as usual, but with that data excluded until you take action. If you would like to check, bsondump the file:

bsondump /data/db/replset/rs_2/rollback/test.rollback.2012-05-28T12-46-19.0.bson

```
{ "_id" : ObjectId( "4fc36da955faa09d71a85ac2" ), "a" : 1992 }
{ "_id" : ObjectId( "4fc36da955faa09d71a85ac3" ), "a" : 1993 }
*snip*
{ "_id" : ObjectId( "4fc3704b55faa09d71a874ca" ), "a" : 8656 }
{ "_id" : ObjectId( "4fc3704b55faa09d71a874cb" ), "a" : 8657 }
6666 objects found
```

Check for the _id in the primary:

```
PRIMARY> db.rollback.find({"_id" : ObjectId( "4fc3704b55faa09d71a874cb" )})
PRIMARY>
```

As expected, the record is not found, not until it is merged back in (assuming that is what you want. As long as you have not inserted more than 300MB of data (at which point an automatic rollback will not be attempted), that is how a rollback happens and how it can be simulated and worked with.

NOTE: With the oplog size of 100MB being set by default in the mongo-snippets script it is not possible to simulate the 300MB+ rollback with the procedure described above. I will use a follow up post to outline that particular case and how it behaves.

Email

Not published

Website

Comment

Post It

Notify me of follow-up comments by email.

Notify me of new posts by email.

Gamification – A New Twist on the Carrot and Stick

MongoDB Melbourne and Sydney