Wednesday, June 19, 2013

MongoDB course note 6

Write Concern

Provided you assume that the disk is persistent, what are the w and j settings required to guarantee that an insert or update has been written all the way to disk.

w=1, j=1

Network Errors

What are the reasons why an application may receive an error back even if the write was successful. Check all that apply.

- ☑ The network TCP network connection between the application and the server was reset between the time of the write and the time of the getLastError call.
- ☑ The MongoDB server terminates between the write and the getLastError call.
- ☑ The network fails between the time of the write and the time of the getLastError call
- □ The write violates a primary key constraint on the collection and must be rolled back.

Introduction to Replication

What is the minimum original number of nodes needed to assure the election of a new Primary if a node goes down?

3

Replica Set Elections

Which types of nodes can participate in elections of a new primary?

- ☑ Regular replica set members
- ☑ Hidden Members
- ☑ Arbiters
- □ Lawyers

Write Consistency

During the time when failover is occurring, can writes successfully complete?

- ○ Yes
- ● No

Creating a Replica Set

Search This Blog

Search

Labels

- Java (48)
- Web Service (14)
- Ruby on Rails (13)
- Hadoop (11)
- SCM (10)
- MVNE / MVNO / Telecom (9)
- Mongodb (9)
- Git (5)
- Rest (5)
- Unit test (5)
- clean code (5)
- Software architecture (4)
- Ruby (3)
- Heroku (2)
- Lunix (2)
- Requirement engineering (2)
- C# (1)
- CSS (1)
- Python (1)

About Me

Copenhagen, Denmark

I'm a software developer, clean code practitioner and requirement analyst. I'm working on improving software quality from software engineering perspectives (Requirement engineering, software architecture, design, SCM, clean code, and testing). Currently, I work as a Telecom technical consultant in CPH.

View my complete profile

Which command, when issued from the mongo shell, will allow you to read from a secondary?

- ○ db.isMaster()
- ○ db.adminCommand({'readPreference':'Secondary'})
- ○ rs.setStatus("Primary")
- ● rs.slaveOk()

Replica Set Internals

In the video how long did it take to elect a new primary?

- ● About three seconds
- ○ Above 10 seconds
- ○ About 30 seconds
- ○ About a minute

Failover and Rollback

What happens if a node comes back up as a secondary after a period of being offline and the oplog has looped on the primary?

The entire dataset will be copied from the primary

Connecting to a Replica Set from Java

If you leave a replica set node out of the seedlist within the driver, what will happen?

- ○ The missing node will not be used by the application.
- ● The missing node will be discovered as long as you list at least one valid node.
- ○ This missing node will be used for reads, but not for writes.
- ○ The missing node will be used for writes, but not for reads.

Bad Things Happen to Good Nodes

If you use the MongoClient constructor that takes a seed list of replica set members, are you guaranteed to avoid application exceptions during a primary failover?

- ○ Yes
- ● No

Introduction to Sharding

If the shard key is not include in a find operation and there are 3 shards, each one a replica set with 3 nodes, how many nodes will see the find operation?

- ○ 1
- ● 3
- ○ 9
- ○ 6

Building a Sharded Environment

If you want to build a production system with two shards, each one a replica set with three nodes, how may mongod processes must you start?

○ 2
○ 6
○ 7
● 9

2 shards has 6 nodes. 3 config nodes

Implications of Sharding

Suppose you wanted to shard the zip code collection after importing it. You want to shard on zip code. What index would be required to allow MongoDB to shard on zip code?

● An index on zip or a non-multi-key index that starts with zip.
○ No index is required to use zip as the shard key.
○ A unique index on the zip code.
○ Any index that that includes the zip code.

Sharding + Replication

Suppose you want to run multiple mongos routers for redundancy. What level of the stack will assure that you can failover to a different mongos from within your application?

○ mongod
○ mongos
● drivers
○ sharding config servers

Choosing a Shard Key

You are building a facebook competitor called footbook that will be a mobile social network of feet. You have decided that your primary data structure for posts to the wall will look like this:

```
{'username':'toeguy',
    'posttime':ISODate("2012-12-02T23:12:23Z"),
    "randomthought": "I am looking at my feet right now",
    'visible_to':['friends','family', 'walkers']}
```

Thinking about the tradeoffs of shard key selection, select the true statements below.

☑ Choosing posttime as the shard key will cause hotspotting as time progresses.
☑ Choosing username as the shard key will distribute posts to the wall well across the shards.
☑ Choosing visible_to as a shard key is illegal.
☐ Choosing posttime as the shard key suffers from low cardinality.

HW 6.1

Which of the following statements are true about MongoDB replication. Check all that apply.

☑ The minimum sensible number of voting nodes to a replica set is three.
☐ MongoDB replication is synchronous.

- □ The Mongo shell is capable of attaching to a replica set and automatically failing over.
- □ By default, using the new MongoClient connection class, w=1 and j=1.
- ☑ The oplog utilizes a capped collection.

HW 6.2

Let's suppose you have a five member replica set and want to assure that writes are committed to the journal and are acknowledged by at least 3 nodes before you proceed forward. What would be the appropriate settings for w and j?

- ○ w=1, j=1
- ● w="majority", j=1
- ○ w=3, j=0
- ○ w=5, j=1
- ○ w=1,j=3

HW 6.3

Which of the following statements are true about choosing and using a shard key:

- □ The shard key must be unique
- ☑ There must be a index on the collection that starts with the shard key.
- ☑ Mongo can not enforce unique indexes on a sharded collection other than the shard key itself.
- ☑ Any update that does not contain the shard key will be sent to all shards.
- □ You can change the shard key on a collection if you desire.

HW 6.4

You have a sharded system with three shards and have sharded the collections "grades" in the "test" database across those shards. The output of sh.status() when connected to mongos looks like this:

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: { "_id" : 1, "version" : 3 }
  shards:
 {  "_id" : "s0",  "host" : "s0/localhost:37017,localhost:37018,localhost:37019" }
 {  "_id" : "s1",  "host" : "s1/localhost:47017,localhost:47018,localhost:47019" }
 {  "_id" : "s2",  "host" : "s2/localhost:57017,localhost:57018,localhost:57019" }
  databases:
 {  "_id" : "admin",  "partitioned" : false,  "primary" : "config" }
 {  "_id" : "test",  "partitioned" : true,  "primary" : "s0" }
  test.grades chunks:
    s1 4
    s0 4
    s2 4
   { "student_id" : { $minKey : 1 } } -->> { "student_id" : 0 } on : s1 Timestamp(12000, 0)
   { "student_id" : 0 } -->> { "student_id" : 2640 } on : s0 Timestamp(11000, 1)
   { "student_id" : 2640 } -->> { "student_id" : 91918 } on : s1 Timestamp(10000, 1)
   { "student_id" : 91918 } -->> { "student_id" : 176201 } on : s0 Timestamp(4000, 2)
   { "student_id" : 176201 } -->> { "student_id" : 256639 } on : s2 Timestamp(12000, 1)
   { "student_id" : 256639 } -->> { "student_id" : 344351 } on : s2 Timestamp(6000, 2)
   { "student_id" : 344351 } -->> { "student_id" : 424983 } on : s0 Timestamp(7000, 2)
   { "student_id" : 424983 } -->> { "student_id" : 509266 } on : s1 Timestamp(8000, 2)
   { "student_id" : 509266 } -->> { "student_id" : 596849 } on : s1 Timestamp(9000, 2)
   { "student_id" : 596849 } -->> { "student_id" : 772260 } on : s0 Timestamp(10000, 2)
   { "student_id" : 772260 } -->> { "student_id" : 945802 } on : s2 Timestamp(11000, 2)
   { "student_id" : 945802 } -->> { "student_id" : { $maxKey : 1 } } on : s2 Timestamp(11000, 3)
```

If you ran the query

```
use test
db.grades.find({'student_id':530289})
```

Which shards would be involved in answering the query?

○ s0,s1 and s2

○ s0

● s1

○ s2

HW 6.5

Create three directories for the three mongod processes. On unix, this could be done as follows:

```
mkdir -p /data/rs1 /data/rs2 /data/rs3
```

Now start three mongo instances as follows. Note that are three commands. The browser is probably wrapping them visually.

```
./mongod --replSet m101 --logpath "1.log" --dbpath /data/rs1 --port 27017 --smallfiles --fork

./mongod --replSet m101 --logpath "2.log" --dbpath /data/rs2 --port 27018 --smallfiles --fork

./mongod --replSet m101 --logpath "3.log" --dbpath /data/rs3 --port 27019 --smallfiles --fork
```

Now connect to a mongo shell and make sure it comes up

```
./mongo --port 27017
```

Now you will create the replica set. Type the following commands into the mongo shell:

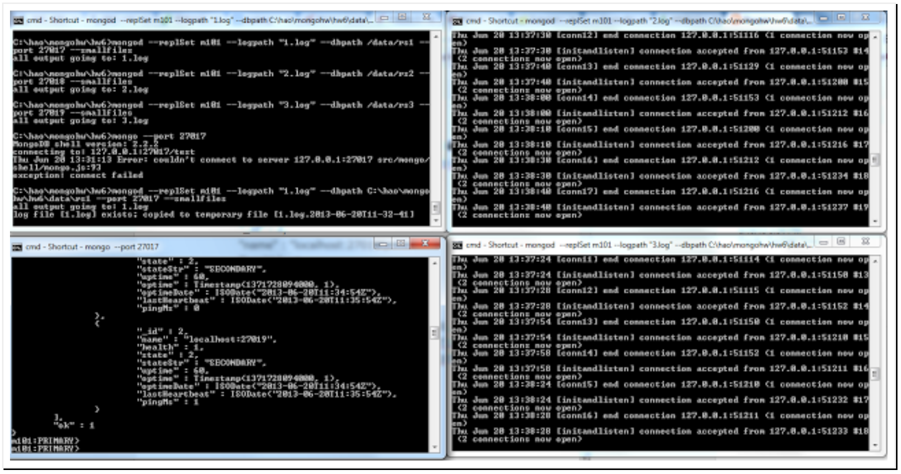
```
config = { _id: "m101", members:[
          { _id : 0, host : "localhost:27017"},
          { _id : 1, host : "localhost:27018"},
          { _id : 2, host : "localhost:27019"} ]
};
rs.initiate(config);
```

At this point, the replica set should be coming up. You can type

```
rs.status()
```

to see the state of replication.

```
m101:PRIMARY> rs.status()
{
        "set" : "m101",
        "date" : ISODate("2013-06-20T11:35:56Z"),
        "myState" : 1,
        "members" : [
                {
                        "_id" : 0,
                        "name" : "localhost:27017",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 195,
                        "optime" : Timestamp(1371728094000, 1),
                        "optimeDate" : ISODate("2013-06-20T11:34:54Z"),
                        "self" : true
                },
                {
                        "_id" : 1,
                        "name" : "localhost:27018",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 60,
                        "optime" : Timestamp(1371728094000, 1),
                        "optimeDate" : ISODate("2013-06-20T11:34:54Z"),
                        "lastHeartbeat" : ISODate("2013-06-20T11:35:54Z"),
                        "pingMs" : 0
                },
                {
                        "_id" : 2,
                        "name" : "localhost:27019",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 60,
                        "optime" : Timestamp(1371728094000, 1),
                        "optimeDate" : ISODate("2013-06-20T11:34:54Z"),
                        "lastHeartbeat" : ISODate("2013-06-20T11:35:54Z"),
                        "pingMs" : 1
                }
        ],
        "ok" : 1
}
```



cmd - Shortcut - mongod --replSet m101 --logpath "1.log" --dbpath C:\hao\mongohw\hw6\data\...

```
C:\hao\mongohw\hw6>mongod --replSet m101 --logpath "1.log" --dbpath /data/rs1 --
port 27017 --smallfiles
all output going to: 1.log

C:\hao\mongohw\hw6>mongod --replSet m101 --logpath "2.log" --dbpath /data/rs2 --
port 27018 --smallfiles
all output going to: 2.log

C:\hao\mongohw\hw6>mongod --replSet m101 --logpath "3.log" --dbpath /data/rs3 --
port 27019 --smallfiles
all output going to: 3.log

C:\hao\mongohw\hw6>mongo --port 27017
MongoDB shell version: 2.2.2
connecting to: 127.0.0.1:27017/test
Thu Jun 20 13:31:13 Error: couldn't connect to server 127.0.0.1:27017 src/mongo/
shell/mongo.js:93
exception: connect failed

C:\hao\mongohw\hw6>mongod --replSet m101 --logpath "1.log" --dbpath C:\hao\mongo
hw\hw6\data\rs1 --port 27017 --smallfiles
all output going to: 1.log
log file [1.log] exists; copied to temporary file [1.log.2013-06-20T11-32-41]
```

cmd - Shortcut - mongod --replSet m101 --logpath "2.log" --dbpath C:\hao\mongohw\hw6\data\...

```
Thu Jun 20 13:37:30 [conn12] end connection 127.0.0.1:51116 (1 connection now op
en)
Thu Jun 20 13:37:30 [initandlisten] connection accepted from 127.0.0.1:51153 #14
 (2 connections now open)
Thu Jun 20 13:37:40 [conn13] end connection 127.0.0.1:51129 (1 connection now op
en)
Thu Jun 20 13:37:40 [initandlisten] connection accepted from 127.0.0.1:51200 #15
 (2 connections now open)
Thu Jun 20 13:38:00 [conn14] end connection 127.0.0.1:51153 (1 connection now op
en)
Thu Jun 20 13:38:00 [initandlisten] connection accepted from 127.0.0.1:51212 #16
 (2 connections now open)
Thu Jun 20 13:38:10 [conn15] end connection 127.0.0.1:51200 (1 connection now op
en)
Thu Jun 20 13:38:10 [initandlisten] connection accepted from 127.0.0.1:51216 #17
 (2 connections now open)
Thu Jun 20 13:38:30 [conn16] end connection 127.0.0.1:51212 (1 connection now op
en)
Thu Jun 20 13:38:30 [initandlisten] connection accepted from 127.0.0.1:51234 #18
 (2 connections now open)
Thu Jun 20 13:38:40 [conn17] end connection 127.0.0.1:51216 (1 connection now op
en)
Thu Jun 20 13:38:40 [initandlisten] connection accepted from 127.0.0.1:51237 #19
 (2 connections now open)
```

```
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 60,
                        "optime" : Timestamp(1371728094000, 1),
                        "optimeDate" : ISODate("2013-06-20T11:34:54Z"),
                        "lastHeartbeat" : ISODate("2013-06-20T11:35:54Z"),
                        "pingMs" : 0
                },
                {
                        "_id" : 2,
                        "name" : "localhost:27019",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 60,
                        "optime" : Timestamp(1371728094000, 1),
                        "optimeDate" : ISODate("2013-06-20T11:34:54Z"),
                        "lastHeartbeat" : ISODate("2013-06-20T11:35:54Z"),
                        "pingMs" : 1
                }
        ],
        "ok" : 1
}
m101:PRIMARY>
m101:PRIMARY>
```

cmd - Shortcut - mongod --replSet m101 --logpath "3.log" --dbpath C:\hao\mongohw\hw6\data\...

```
Thu Jun 20 13:37:24 [conn11] end connection 127.0.0.1:51114 (1 connection now op
en)
Thu Jun 20 13:37:24 [initandlisten] connection accepted from 127.0.0.1:51150 #13
 (2 connections now open)
Thu Jun 20 13:37:28 [conn12] end connection 127.0.0.1:51115 (1 connection now op
en)
Thu Jun 20 13:37:28 [initandlisten] connection accepted from 127.0.0.1:51152 #14
 (2 connections now open)
Thu Jun 20 13:37:54 [conn13] end connection 127.0.0.1:51150 (1 connection now op
en)
Thu Jun 20 13:37:54 [initandlisten] connection accepted from 127.0.0.1:51210 #15
 (2 connections now open)
Thu Jun 20 13:37:58 [conn14] end connection 127.0.0.1:51152 (1 connection now op
en)
Thu Jun 20 13:37:58 [initandlisten] connection accepted from 127.0.0.1:51211 #16
 (2 connections now open)
Thu Jun 20 13:38:24 [conn15] end connection 127.0.0.1:51210 (1 connection now op
en)
Thu Jun 20 13:38:24 [initandlisten] connection accepted from 127.0.0.1:51232 #17
 (2 connections now open)
Thu Jun 20 13:38:28 [conn16] end connection 127.0.0.1:51211 (1 connection now op
en)
Thu Jun 20 13:38:28 [initandlisten] connection accepted from 127.0.0.1:51233 #18
 (2 connections now open)
```

Posted by . at 3:34 PM

+1 Recommend this on Google

Labels: Mongodb

2 comments:

dandy February 17, 2014 at 12:39 AM

what is the answer after doing python validate.py?

Reply

Unknown March 12, 2014 at 11:07 AM

@dandy this is the code - kjvjkl3290mf0m20f2kjjv

Reply

Enter your comment...

Comment as: Google Accou ▼

Publish Preview

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Blog Archive

- ▼ 2013 (85)
  - ► November (1)
  - ► September (18)
  - ► August (6)
  - ► July (13)
  - ▼ June (11)
    - Better Rest API design 4
    - Better Rest API design 3
    - Better Rest API design 2
    - Better Rest API design 1
    - Liferay (Tomcat container) setup connections pool
    - MongoDB course note 7 (final)
    - ssh and scp
    - Aws, git, heroku env setup
    - MongoDB course note 6
    - MongoDB course note 5
    - MongoDB course note 4
  - ► May (14)
  - ► March (18)
  - ► February (4)
- ► 2012 (8)
- ► 2011 (37)
- ► 2010 (19)