Mongoshake 를 이용한 양방향 실시간 동기화 성공 / 테스트

간단 요약 :

단방향 동기화를 2 way 로 구성하면서 양방향 동기화를 성공한 듯했다. 하지만 실시간으로 돌리려고 했을 때 동시에는 진행이 되지 않았던 것이 있었다.

문제를 파악하기 위해

- Mongoshake English document
- Mongoshake FAQ document
- Mongoshake MongoShake 最佳实践

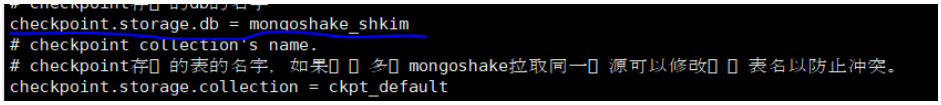
등을 뒤져보았다.... 하지만 정확한 설명이나 나와 같은 error 에 대한 설명이 없었고 조금 더 제대로 파악을 하기 위해 collector.conf 파일의 옵션들을 연구해보았다.

mongoshake 은 앞서 말했던 것 처럼 oplog 를 보고 sync 를 진행한다. 또 checkpoint DB 를 생성 (default : mongoshake.ckpt_default) 하여 이곳에 시간을 기준으로 적는다.

이 check point 를 기준으로 이후 sync 가 진행 되게 되는 것이다.

현재 실시간으로 프로세스를 양쪽 서버에서 돌리게 생성된 checkpoint DB 를 양쪽에서 서로 동기화를 하면서 꼬이는 것 같다는 생각을 하게 되었다.

1. 첫번째 시도한 방법은 양쪽 서버의 default checkpoint DB 를 바꾸는 것이다.



위와 같이 서울 리전의 서버에서는 mongoshake_shyunkim / 싱가포르 리전에서는 mongoshake_shkim 으로 진행해 보았다.

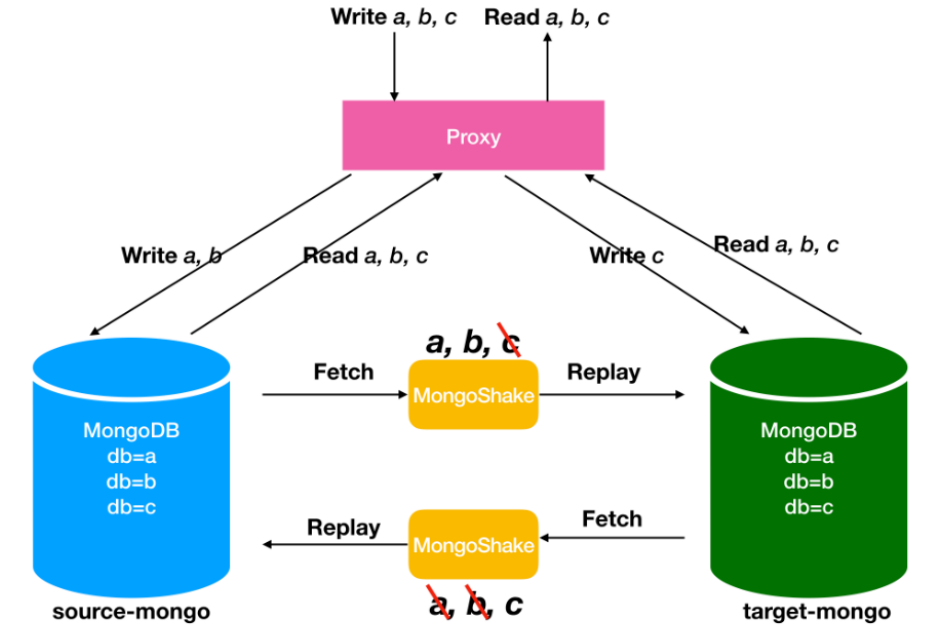
하지만 같은 이유의 error 가 출력되면서 양쪽 동시에 진행되지 않았다.

다른 방법을 찾아보았다.

mongoshake 공식 문서에서 찾은 내용이다.

Q: How to build active-active replication(双活) in the current opensource version without `gid` support?

A: User can use filter(`filter.namespace.white` and `filter.namespace.black`) to fulfill this function. Currently, the granularity of filter is collection.
For example, I have three databases in one mongodb replicaSet named `a` , `b` and `c` . Assume source replicaSet is `source-mongo` and target replicaSet is `target-mongo` , so we build two MongoShakes to fetch oplog from `source-mongo` and `target-mongo` respectively. And in the first MongoShake we only filter database name is equal to `a` or `b` while in the second MongoShake we only filter `c` . I draw a figure to make explanation clearly.



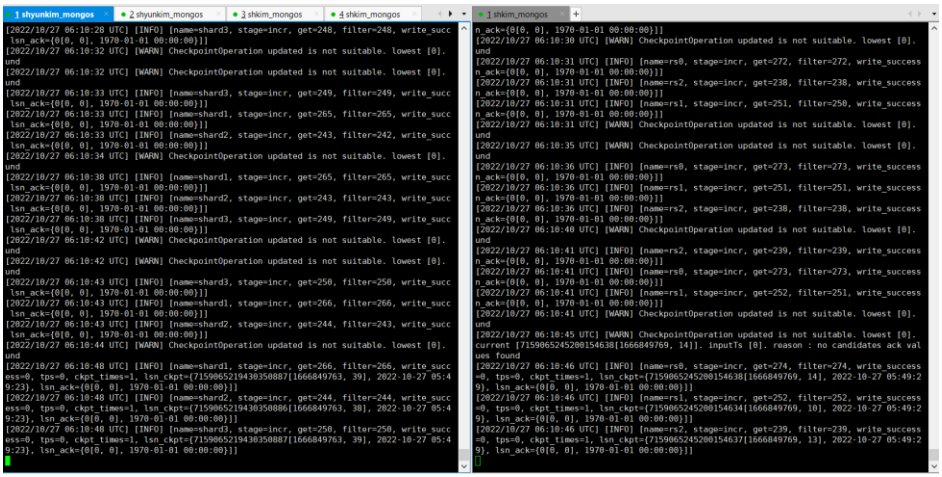
Users can use their own `proxy` program to distribute data so that writing of `a` and `b` will go to the source database while `c` goto the target.

정확히 내가 찾는 내용과는 거리가 있지만 중요한것은 mongoshake 은 동기화 진행되는 collection , DB 를 사용자가 직접 정해줄 수 있다.

filter.namespace.black 옵션을 통해 특정 컬렉션을 동기화에서 빼낼 수 있고 이를 이용해 checkpoint DB 를 제외한 상태로 sync 가 이루어질 수 있도록 옵션을 수정했다.

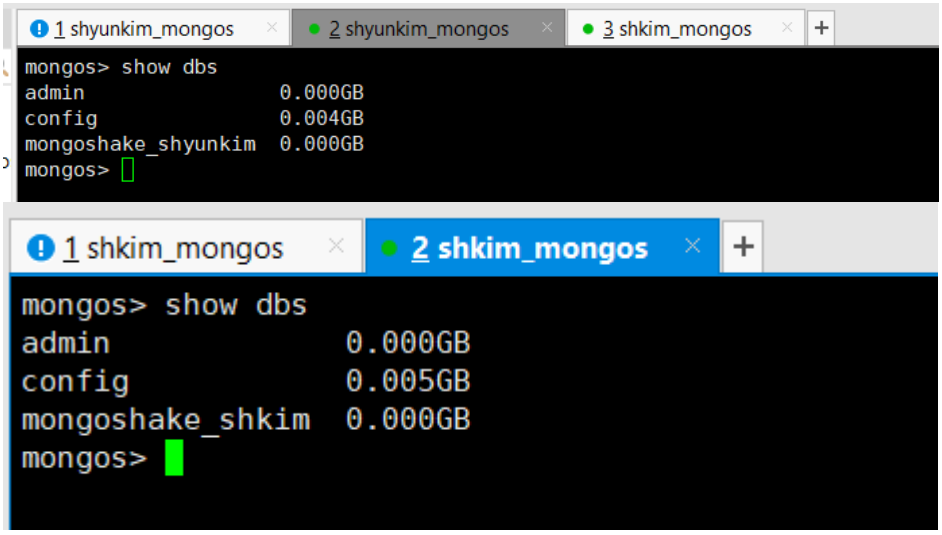
```
# 不能同□ 指定。分□ 分割不同namespace，每□ namespace可以是db，也可
filter.namespace.black = mongoshake_shyunkim.ckpt_default
filter.namespace.white =
```

여기서 한가지 주의해야할 점은 src 에서의 checkpoint DB 가 아닌 dst 에서의 checkpoint DB 를 넣어주어야한다는 것이다.



양쪽 서버에서 동시에 운영되고 있는 모습을 볼 수 있다.

이젠 테스트다. 현재 양쪽 mongos 서버의 DB 내역이다



각각 checkpointDB 가 존재하고 기본적으로 admin / config 등의 DB 는 default 로 동기화에서 제외되기 때문에 현재는 동기화 할 DB 가 없는 셈이다.

싱가포르 리전쪽 클러스터에 10 만개의 더미데이터를 생성해 보겠다.

```
switched to db test
mongos> for (i =0;i<100000;i++) db.item.insertOne({"num":i})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("635a23bd91ddaefd0117ee83")
}
mongos>
```

```
mongos> db.item.find().count()
100000
mongos>
```

```
mongos> db.item.find().count()
100000
mongos>
```

보다시피 양쪽 서버 모두 동시에 동기화가 완료된 모습이다.

반대 편에서도 10 만개를 삽입했고 동기화되는 것을 확인했다.

```
mongos> db.item.find().count()
173976
mongos> for(i =100000;i<200000;i++) db.item.insertOne({"num":i})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("635a31efab7dffc80011e331")
}
mongos> db.item.find().count()
273976
```

```
mongos> db.item.find().count()
177815
mongos> db.item.find().count()
273976
```

** 이 상태에서 한쪽 클러스터에서 sharding 을 하게 되면 어떻게 적용되는지 궁금하여 급하게 서울 리전(shyunkim_mongos) 쪽의 test DB 의 item collection 을 3 shard 로 구성해보았다.*

→ 샤딩은 되지 않았다..! 할

사실 조금만 생각해보면 당연한 것이다. 클러스터의 중요 config DB 등은 동기화가 되지 않는다. 그 뜻은 즉 shard 관련 설정 등 또한 동기화 되지 않는 다는 것이다.

이번에는 서울 리전쪽 클러스터에서 10 만개의 더미데이터를 생성해보겠다.

```
mongos> for(i =100000;i<200000;i++) db.item.insertOne({"num":i})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("635a2918ab7dffc800105c91")
}
```

당연히 20 만개가 되어야한다.

```
mongos> db.item.getShardDistribution()

Shard shard1 at shard1/172.32.130.55:27017,172.32.176.183:27017,172.32.82.245:27017
 data : 6.24MiB docs : 187172 chunks : 1
 estimated data per chunk : 6.24MiB
 estimated docs per chunk : 187172

Shard shard3 at shard3/172.32.162.46:27017,172.32.42.154:27017,172.32.51.237:27017
 data : 3.33MiB docs : 100058 chunks : 1
 estimated data per chunk : 3.33MiB
 estimated docs per chunk : 100058

Totals
 data : 9.58MiB docs : 287230 chunks : 2
 Shard shard1 contains 65.16% data, 65.16% docs in cluster, avg obj size on shard : 35B
 Shard shard3 contains 34.83% data, 34.83% docs in cluster, avg obj size on shard : 35B

mongos> db.item.find().count()
287230
```

..?

시간이 지나면 알아서 샤딩이 완료되고 정상적으로 되는 경우가 많아 기다려보았지만 변화가 없길래

mongoshake 서버를 조회해보았더니 서울 리전의 mongoshake 가 에러로 꺼진 듯 했다.

간단히 원인을 파악해 보자면 새로 생긴 collection 의 sharding 을 했더니 자동으로 balancer 가 켜진 것이고 mongoshake 는 balancer 가 꺼져야 한다.

```
mongos> sh.disableBalancing(
... "test.item")
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

바로 balancer 를 끄고 다시 실행하였다

하지만 에러가 사라지지 않는다..

다른 추측으로는 한쪽으로는 sharding 이 진행된 컬렉션이고 다른 쪽은 sharding 이 되지 않은 컬렉션이기 때문에 그런 걸까..?

오류 해결을 위해 sharding 설정했던 DB 를 삭제하고 삭제가 동기화 되기를 기다렸지만 진행되지 않았다.

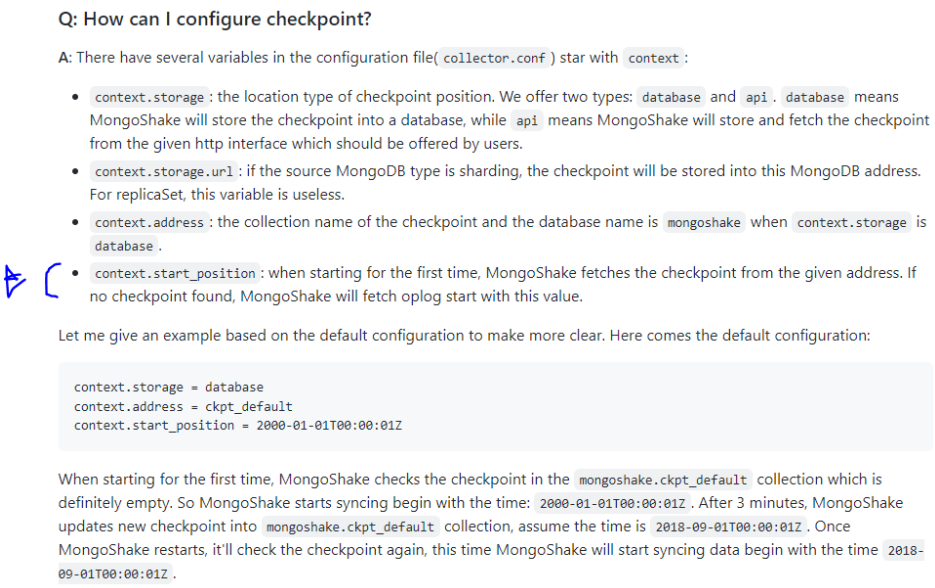
이는 mongoshake 의 기본적인 원리 때문인데 mongoshake 는 checkpoint DB 가 정상적으로 존재할 때는 full sync 가 아닌 incr sync 모드로 작동한다.

(full 과 incr 의 차이는 기본적으로 full 은 말그대로 모든 데이터를 다시 동기화 하는 것이고 incr 은 계속 추가해 나가는 것이다.)

그럼 한 쪽에서 데이터베이스를 삭제하고 나면 checkpoint 를 새롭게 지정해주어야한다.

본인은 checkpoint DB 를 삭제하고 다시 full sync 를 진행함으로써 다시금 동기화 시켰다.

현업에서는 이러한 방법을 사용하기 어려울 수 있다. 이때 사용할 수있는 방법이



Q: How can I configure checkpoint?

A: There have several variables in the configuration file(`collector.conf`) star with `context` :

- `context.storage` : the location type of checkpoint position. We offer two types: `database` and `api` . `database` means MongoShake will store the checkpoint into a database, while `api` means MongoShake will store and fetch the checkpoint from the given http interface which should be offered by users.
- `context.storage.url` : if the source MongoDB type is sharding, the checkpoint will be stored into this MongoDB address. For replicaSet, this variable is useless.
- `context.address` : the collection name of the checkpoint and the database name is `mongoshake` when `context.storage` is `database` .
- `context.start_position` : when starting for the first time, MongoShake fetches the checkpoint from the given address. If no checkpoint found, MongoShake will fetch oplog start with this value.

Let me give an example based on the default configuration to make more clear. Here comes the default configuration:

```
context.storage = database
context.address = ckpt_default
context.start_position = 2000-01-01T00:00:01Z
```

When starting for the first time, MongoShake checks the checkpoint in the `mongoshake.ckpt_default` collection which is definitely empty. So MongoShake starts syncing begin with the time: `2000-01-01T00:00:01Z` . After 3 minutes, MongoShake updates new checkpoint into `mongoshake.ckpt_default` collection, assume the time is `2018-09-01T00:00:01Z` . Once MongoShake restarts, it'll check the checkpoint again, this time MongoShake will start syncing data begin with the time `2018-09-01T00:00:01Z` .

이 방법이다. context.start_position 옵션을 활용하면 checkpoint 위치를 조정할 수 있다.