Gestión de grandes volúmenes de datos

Trabajo Práctico de sharding

Jose Mauricio Lara Tapia

Partiendo de la base del trabajo de réplica set ya tendremos las rutas creadas, en mi caso el sistema operativo utilizado es Ubuntu 20.04 LTS. Se dispondrá de 3 nodos, uno primario y dos secundarios.

- /home/mauri/data/db/rs/0
- /home/mauri/data/db/rs/1
- /home/mauri/data/db/rs/2

Ejecutamos los siguientes comandos en 3 terminales separadas

```
mongod --shardsvr --replSet rs0 --dbpath /home/mauri/data/db/rs/0 --port 27060
mongod --shardsvr --replSet rs0 --dbpath /home/mauri/data/db/rs/1 --port 27061
mongod --shardsvr --replSet rs0 --dbpath /home/mauri/data/db/rs/2 --port 27062
```

En el shell de mongo configuramos el primer nodo del replica set

```
mongo --port 27060
```

```
cfg = {
        _id:"rs0",
        members:[
                {_id:0, host:"localhost:27060"},
                {_id:1, host:"localhost:27061"},
                {_id:2, host:"localhost:27062"}
            ]
        };
```

```
rs.initiate(cfg)
```

```
> cfg = {
... _id:"rs0",
... members:[
... {_id:0, host:"localhost:27060"},
... {_id:1, host:"localhost:27061"},
... {_id:2, host:"localhost:27062"}
... ]
... };
{
        "_id" : "rs0",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "localhost:27060"
                },
                {
                        "_id" : 1,
                        "host" : "localhost:27061"
                },
                {
                        "_id" : 2,
                        "host" : "localhost:27062"
                }
        ]
}
> rs.initiate(cfg)
{ "ok" : 1 }
rs0:SECONDARY>
```

Configuramos el segundo nodo

mongo --port 27061

rs.secondaryOk()

use finanzas

```
rs0:SECONDARY> rs.secondaryOk()
rs0:SECONDARY> use finanzas
switched to db finanzas
rs0:SECONDARY>
```

Por último configuramos el tercer nodo

mongo --port 27062

rs.secondaryOk()

use finanzas

Ahora debemos levantar el config server configurandolo como replica set con un solo nodo por el momento. Para ello en la carpeta /home/mauri/data/configsvr levantamos el nodo.

mongod --dbpath /home/mauri/data/configsvr --configsvr --replSet c --port 27500

y en otra terminal realizamos la configuración del nodo

mongo --port 27500

```
rs.initiate({
                _id: "c",members:[{
                                _id: 0,
                                host: "localhost:27500"
                        }]
        })
```

chequeamos el estado del nodo con el comando

rs.status()

```
c:SECONDARY> rs.status()
{
        "set" : "c",
        "date" : ISODate("2021-09-28T14:50:19.694Z"),
        "myState" : 1,
        "term" : NumberLong(1),
        "syncSourceHost" : "",
        "syncSourceId" : -1,
        "configsvr" : true,
        "heartbeatIntervalMillis" : NumberLong(2000),
        "majorityVoteCount" : 1,
        "writeMajorityCount" : 1,
        "votingMembersCount" : 1,
        "writableVotingMembersCount" : 1,
        "optimes" : {
                "lastCommittedOpTime" : {
                        "ts" : Timestamp(1632840619, 2),
                        "t" : NumberLong(1)
                },
                "lastCommittedWallTime" : ISODate("2021-09-28T14:50:19.307Z"),
                "readConcernMajorityOpTime" : {
                        "ts" : Timestamp(1632840619, 2),
                        "t" : NumberLong(1)
                },
                "readConcernMajorityWallTime" : ISODate("2021-09-28T14:50:19.307Z"),
                "appliedOpTime" : {
                        "ts" : Timestamp(1632840619, 2),
                        "t" : NumberLong(1)
                },
                "durableOpTime" : {
                        "ts" : Timestamp(1632840619, 2),
                        "t" : NumberLong(1)
                },
                "lastAppliedWallTime" : ISODate("2021-09-28T14:50:19.307Z"),
                "lastDurableWallTime" : ISODate("2021-09-28T14:50:19.307Z")
        },
        "lastStableRecoveryTimestamp" : Timestamp(1632840598, 1),
        "electionCandidateMetrics" : {
                "lastElectionReason" : "electionTimeout",
                "lastElectionDate" : ISODate("2021-09-28T14:48:58.008Z"),
                "electionTerm" : NumberLong(1),
                "lastCommittedOpTimeAtElection" : {
                        "ts" : Timestamp(0, 0),
                        "t" : NumberLong(-1)
                },
                "lastSeenOpTimeAtElection" : {
                        "ts" : Timestamp(1632840537, 1),
                        "t" : NumberLong(-1)
                },
                "numVotesNeeded" : 1,
                "priorityAtElection" : 1,
                "electionTimeoutMillis" : NumberLong(10000),
                "newTermStartDate" : ISODate("2021-09-28T14:48:58.048Z"),
                "wMajorityWriteAvailabilityDate" : ISODate("2021-09-28T14:48:58.268Z")
        },
        "members" : [
```

```
                {
                        "_id" : 0,
                        "name" : "localhost:27500",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 151,
                        "optime" : {
                                "ts" : Timestamp(1632840619, 2),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2021-09-28T14:50:19Z"),
                        "syncSourceHost" : "",
                        "syncSourceId" : -1,
                        "infoMessage" : "Could not find member to sync from",
                        "electionTime" : Timestamp(1632840538, 1),
                        "electionDate" : ISODate("2021-09-28T14:48:58Z"),
                        "configVersion" : 1,
                        "configTerm" : 1,
                        "self" : true,
                        "lastHeartbeatMessage" : ""
                }
        ],
        "ok" : 1,
        "$gleStats" : {
                "lastOpTime" : Timestamp(1632840537, 1),
                "electionId" : ObjectId("7fffffff0000000000000001")
        },
        "lastCommittedOpTime" : Timestamp(1632840619, 2),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1632840619, 2),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1632840619, 2)
}
c:PRIMARY>
```

Ahora levantamos el servidor de ruteo mongos

mongos --configdb c/localhost:27500

Nos conectamos a mongo para configurar el shard

mongo

sh.addShard("rs0/localhost:27060")

use finanzas

se crea el índice

db.facturas.createIndex({"cliente.region": 1, condPago: 1, _id: 1})

Se activa el shardin para la base de datos finanzas

sh.enableSharding("finanzas")

Hacemos el sharding para la colección facturas de la base de datos finanzas

sh.shardCollection("finanzas.facturas", {"cliente.region": 1, condPago: 1, _id: 1})

y vemos el estado con

sh.status()

use config
db.chunks.find({}, {min:1,max:1,shard:1,_id:0,ns:1}).pretty()

para ver el chunk, aunque por el momento está vacío.

Para generar datos en el replica set en el nodo primario ejecutamos

```
for (var i=0; i<5; i++) {
   load("facts.js")
}
```

Chequeamos nuevamente el estado del chunk

sh.status()

use config

db.chunks.find({}, {min:1,max:1,shard:1,_id:0,ns:1}).pretty()

y ahora sí se puede observar datos en los chunks.

Las consultas a ejecutar son:

db.facturas.find({"cliente.region":"CABA", "condPago":"30 Ds FF"}).explain()

db.facturas.find({"cliente.apellido":"Manoni"}).explain()

Agregamos un nuevo shard

mongod --shardsvr --replSet rs20 --dbpath /home/mauri/data/db/rs/20 --port 29060
mongod --shardsvr --replSet rs20 --dbpath /home/mauri/data/db/rs/21 --port 29061
mongod --shardsvr --replSet rs20 --dbpath /home/mauri/data/db/rs/22 --port 29062
mongo --port 29060

al igual que hicimos anteriormente lo configuramos con:

```
cfg = {
        _id:"rs20",
        members:[
                {_id:0, host:"localhost:29060"},
                {_id:1, host:"localhost:29061"},
                {_id:2, host:"localhost:29062"}
        ]
};

rs.initiate(cfg)
```

mongo --port 29061

rs.secondaryOk()

use finanzas

mongo --port 29062

rs.secondaryOk()

use finanzas

Por último, en mongo agregamos el nuevo shard:

sh.addShard("rs20/localhost:29060")

Y vemos el nuevo estado del cluster con

#sh.status()