Bases de Datos a Grande Escala
Entrega 4

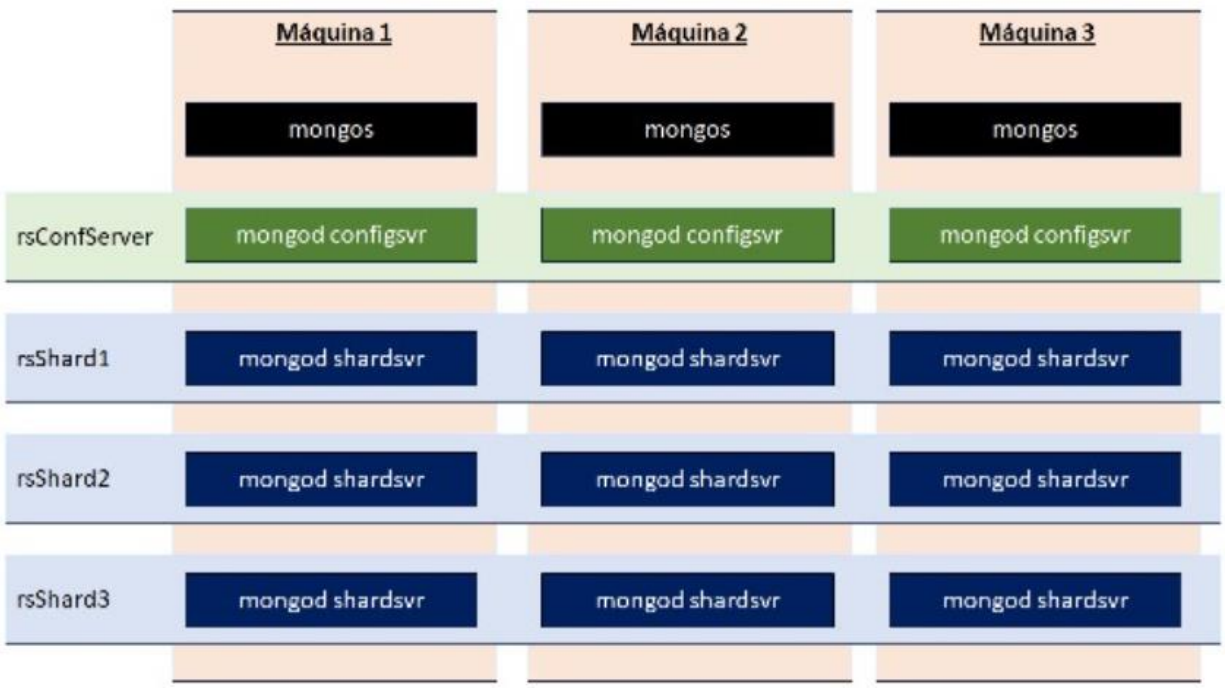
ANDRÉS CAMPOS CUIÑA

FECHA DE ENTREGA: 10/12/2021

ÍNDICE

1 INTRODUCCIÓN

En primer lugar, lo que haremos en esta práctica es crear un clúster de MongoDB con la siguiente arquitectura:



2 PASO 1

Lo primero que debemos hacer es clonar las tres máquinas a partir de la máquina **greiBDServerBase**. Esta máquina ya tiene MongoDB, pero es necesario generar nuevas direcciones MAC para los adaptadores de red, cambiar el nombre del host en el sistema operativo y cambiar el identificador de la máquina en el sistema operativo.

Estos tres pasos ya fueron realizados para llevar a cabo la Práctica 3 al montar el clúster de CITUS, por lo que no se tienen que volver a llevar a cabo de nuevo.

Estas máquinas reciben los nombres **cluster1**,**cluster2** y **cluster3**.

3 PASO 2

Para iniciar el servidor de configuración repetimos los siguientes pasos. En primer lugar, debemos abrir una terminal y crear una carpeta para sus datos. Después iniciamos el proceso **mongod configsvr**. Para esto hacemos uso de los siguientes comandos:

```
mkdir servconf

nohup mongod --configsvr --replSet rsConfServer --port 27017 --dbpath
/home/alumnogreibd/servconf --bind_ip localhost,192.168.56.xxx > /dev/null &
```

Mediante este comando le asignamos el nombre **rsConfServer** a nuestro replica set mediante la opción **–replSet**, la ruta a la carpeta de datos mediante la opción **–dbpath** y la opción **–bind_ip** para definir la dirección IP que se usará para lanzar peticiones al nodo. Como cada una de las instancias en este proceso está en una máquina distinta, debemos usar 'localhost,192.168.56.xxx' con la dirección IP apropiada en cada una de las máquinas, para admitir conexiones tanto desde localhost como desde cualquier otra de las máquinas.

Finalmente, redireccionamos la salida a **/dev/null** para que no nos moleste en la terminal y terminamos el comando con el símbolo **&** para que quede funcionando en background.

Esto debemos llevarlo a cabo en cada una de las tres máquinas, estableciendo la dirección IP correspondiente en cada una de ellas.

El siguiente paso será configurar el replicaSet usando la función **rs.initiate**. Para esto debemos conectarnos desde sólo una de las máquinas (en este caso se hace desde la máquina **cluster1)** a Mongo para iniciar el replicaSet mediante el siguiente comando:

```
mongo --host localhost --port 27017
```

Dentro iniciamos el replicaSet con la función **rs.initiate**. Ahora debemos ejecutar esta función con el siguiente contenido, para tener en cuenta a todos los miembros del replicaSet:

```
rs.initiate({
  _id: "rsConfServer",
  configsvr: true,
  members: [
      {_id: 0, host: "192.168.56.105:27017"},
      {_id: 1, host: "192.168.56.106:27017"},
      {_id: 2, host: "192.168.56.107:27017"}
      ]
})
```

Tras ejecutar este función ya tenemos iniciado el replicaSet "rsConfServer".

4 PASO 3

Ahora iniciaremos un replicaSet para cada uno de los tres shards (**rsShard1**, **rsShard2** y **rsShard3**). Cada uno de los tres replicaSets tendrá tres miembros, uno en cada una de las tres máquinas.

Para cada uno de los tres shards debemos seguir los siguiente pasos para crear el replicaSet de ese shard. En primer lugar debemos crear una carpeta para guardar los datos de ese shard (a esta

carpeta le llamaremos **shardX/** siendo X el número de shard correspondiente) y después debemos iniciar el proceso **mongod shardsvr**.

Para el shard1 esto lo haríamos mediante los siguiente comandos (la ejecución para el shard2 y el shard3 sería análoga):

```
mkdir shard1

nohup mongod --shardsvr --replSet rsShard1 --port 27018 --dbpath
/home/alumnogreibd/shard1 --bind_ip localhost,192.168.56.xxx > /dev/null &
```

En cada una de las tres máquinas, mediante la opción **–bind_ip** debemos establecer la dirección IP de esa máquina tal y como se hizo en el paso anterior. Mediante la opción **–port** se establece el puerto en el que se iniciará el proceso, para cada uno de los shards debemos iniciar estos procesos en puertos diferentes (en este caso se usará el puerto **27018** para el shard1, el puerto **27019** para el shard2 y el puerto **27020** para el shard3).

Una vez que ya hemos llevado este paso a cabo en cada una de las tres máquinas, debemos configurar este replicaSet mediante el uso de la función **rs.initiate**. Para esto nos conectamos a uno de los miembros del replicaSet (en este caso será el **cluster1**) mediante el siguiente comando:

```
mongo --host localhost --port 27018
```

Dentro iniciamos el replicaSet con la función **rs.initiate**. Ahora debemos ejecutar esta función con el siguiente contenido, para tener en cuenta a todos los miembros del replicaSet:

```
rs.initiate({
  _id: "rsShard1",
  members: [
      {_id: 0, host: "192.168.56.105:27018"},
      {_id: 1, host: "192.168.56.106:27018"},
      {_id: 2, host: "192.168.56.107:27018"}
      ]
})
```

Tras ejecutar este función ya tenemos iniciado el replicaSet “rsShard1” y debemos repetir los pasos descritos anteriormente en cada una de las tres máquinas para iniciar también los replicaSets “rsShard2” y “rsShard3”.

5 PASO 4

Ahora iniciamos una instancia de mongos en cada una de las tres máquinas mediante el siguiente comando:

```
nohup mongos --port 27021 --configdb
rsConfServer/192.168.56.105:27017,192.168.56.106:27017,192.168.56.107:27017 --
bind_ip localhost,192.168.56.xxx > /dev/null &
```

Igual que en los pasos anteriores, debemos especificar para la opción **–bind_ip** la dirección IP correspondiente a cada una de las tres máquinas.

Después nos conectamos en uno de las instancias (en nuestro caso al **cluster1**) mediante el comando:

```
mongo --host localhost --port 27021
```

Y aquí añadimos los tres shards ejecutando los siguientes comandos:

```
sh.addShard("rsShard1/192.168.56.105:27018,192.168.56.106:27018,192.168.56.107
:27018")
sh.addShard("rsShard2/192.168.56.105:27019,192.168.56.106:27019,192.168.56.107
:27019")
sh.addShard("rsShard3/192.168.56.105:27020,192.168.56.106:27020,192.168.56.107
:27020")
```

6 PASO 5

En este paso debemos cargar los datos completos de la base de datos películas, usando el archivo JSON generado durante esta práctica. Para esto movemos el archivo **películas.json** de la máquina "GreiBD" a la máquina "cluster1".

Después, cargamos los datos ejecutando el comando siguiente:

```
mongoimport --host=localhost:27021 --db=bdge --collection peliculas --
file=/home/alumnogreibd/data/peliculas.json
```

Entramos al proceso mongo en **cluster1** ejecutando de nuevo:

```
mongo --host localhost --port 27021
```

Una vez dentro ejecutamos el siguiente código para habilitar el sharding en esta base de datos:

```
Use bdge
sh.enableSharding("bdge")
```

7 PASO 6

Ahora indexamos y particionamos la colección utilizando hashing sobre el atributo id mediante la ejecución de los siguientes comandos:

```
db.peliculas.createIndex({id:"hashed"})
sh.shardCollection("bdge.peliculas", {id:"hashed"})
```

8 PASO 7

Ejecutamos la siguiente consulta, que nos devuelve el número de películas que tienen un presupuesto mayor que medio millón de dólares:

```
db.peliculas.find(
  {
    presupuesto:{$gt:500000}
    }
).count()
```

Esta consulta nos devuelve que 7705 películas tienen un presupuesto mayor.

9 PASO 8

Ahora pausamos la máquina **cluster3** y comprobamos, volviendo a ejecutar la consulta anterior, que aún nos devuelve el resultado correcto (debemos tener activada la opción “primaryPreferred” para cambiar la preferencia de lectura para leer, con preferencia del primario, pero si no puede permitimos leer de cualquier otro).

10 PASO 9

Pausamos ahora la máquina **cluster2** y comprobamos que ya no podemos resolver la consulta anterior.

```
abase. Read and write access to data and configuration is unrestricted
---
mongos> use bdge
switched to db bdge
mongos> db.peliculas.find(   {     presupuesto:{$gt:500000}     } ).count()
uncaught exception: Error: count failed: {
        "ok" : 0,
        "errmsg" : "failed on: rsShard2 :: caused by :: Could not find host matc
hing read preference { mode: \"primary\" } for set rsShard2",
        "code" : 133,
        "codeName" : "FailedToSatisfyReadPreference",
        "operationTime" : Timestamp(1637768681, 5),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1637768681, 5),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
} :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
DBQuery.prototype.count@src/mongo/shell/query.js:386:11
@(shell):1:1
mongos>
```

11 PASO 10

Volvemos a iniciar las dos máquina apagadas y volvemos a lanzar los procesos necesarios en cada una de ellas.

Volvemos a ejecutar la consulta y comprobamos que ahora nos vuelve a devolver el resultado correcto:

```
alumnogreibd@cluster1:~$ mongo --host localhost --port 27021
MongoDB shell version v4.4.3
connecting to: mongodb://localhost:27021/?compressors=disabled&gssapiServiceName
Implicit session: session { "id" : UUID("c508c8f4-a78b-473b-90d4-8ca03e51224d")
MongoDB server version: 4.4.3
---
The server generated these startup warnings when booting:
        2021-11-24T12:22:46.940+00:00: Access control is not enabled for the dat
---
mongos> use bdge
switched to db bdge
mongos> db.peliculas.find(
...     {
...       presupuesto:{$gt:500000}
...       }
... ).count()
7705
mongos>
```

12 PASO 11

Modifica la columna "fecha_emision" para que en lugar de ser de tipo string sea de tipo Date. Utiliza para ello el operador "$dateFromString" dentro de una llamada "updateMany":

```
db.peliculas.updateMany(
  {},
  [
    {
    "$set": {
      "fecha_emision": {
        "$dateFromString": {
          dateString: "$fecha_emision"
          }
        }
      }
    }
  ]
);
```

13 PASO 12

Realiza una consulta en la que obtengas todos los títulos y presupuestos de las películas del género "Science Fiction", que tengan unos ingresos de más de 800000000 dolares.

```
db.peliculas.find(
  {
    "generos.nombre": "Science Fiction",
    ingresos:{$gt:800000000}
  },
  {
    titulo:1,
    presupuesto:1,
    _id:0
  }
).pretty()
```

14 PASO 13

Realiza una consulta que obtenga el título y el título original de las películas protagonizadas por "Penélope Cruz" del género de acción (Action). Ordena el resultado por fecha de emisión.

```
db.peliculas.aggregate(
  [
    {
      $match: {
        "reparto.persona.nombre": "Penélope Cruz",
        "generos.nombre": "Action"
      }
    },
    {
      $project: {
        titulo:1,
        titulo_original:1,
        _id:0
      }
    },
    {$sort: {fecha_emision:-1}}
  ]
)
```

15 PASO 14

Realiza una agregación en la que obtengas para cada uno de los géneros disponibles en la base de datos, el número de películas que tiene, el total del presupuesto invertido, el total de ingresos generados y la diferencia entre los ingresos y el presupuesto. Ordena el resultado por esta diferencia.

```
db.peliculas.aggregate([
  { $unwind: '$generos' },
  {
    $replaceRoot: {
        newRoot: {
            $mergeObjects: [
              { ingresos: '$ingresos' },
              { presupuesto: '$presupuesto' },
              "$generos"
            ]
        }
    }
  },
  {
    $group: {
        _id: '$nombre',
        peliculas: { $sum: 1 },
        ingresos_totales: { $sum: '$ingresos' },
        presupuesto_total: { $sum: '$presupuesto' }
    }
  },
  {
    "$project" : {
      _id:0,
      'genero': '$_id',
      'peliculas': '$peliculas',
      'ingresos_totales' : '$ingresos_totales',
      'presupuesto_total' : '$presupuesto_total',
      'beneficios':{'$subtract':['$ingresos_totales', '$presupuesto_total']},
    }
  },
  { $sort: {beneficios:-1} }
])
```

16 PASO 15

Realiza una agregación para obtener los 10 directores que tiene la duración media de sus películas más alta, de entre sus películas con ingresos (ingresos>0). Para cada director, saca el

número total de películas, la duración media de las películas y la suma de los ingresos de las mismas.

```
db.peliculas.aggregate([
  { $unwind: '$personal' },
  {
    $replaceRoot: {
        newRoot: {
            $mergeObjects: [
              { ingresos: '$ingresos' },
              { duracion: '$duracion' },
              "$personal"
            ]
        }
    }
  },
  {
    $match: {
        trabajo: "Director",
        ingresos:{$gt:0}
    }
  },
  {
    $group: {
        _id: '$persona.nombre',
        peliculas: { $sum: 1 },
        ingresos_totales: { $sum: '$ingresos' },
        duracion_media: { $avg: '$duracion' }
    }
  },
  {
    "$project" : {
      _id:0,
      'director': '$_id',
      'peliculas': '$peliculas',
      'ingresos_totales' : '$ingresos_totales',
      'duracion_media' : '$duracion_media'
    }
  },
  { $sort: {duracion_media:-1} },
  { $limit: 10 }
])
```