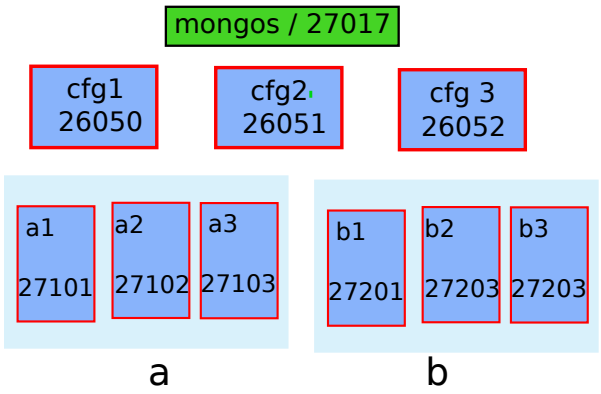
Práctica 8 – Particiones en MongoDB - Solución

Nuestro objetivo es crear un clúster en modo local con:

- 3 servidores de configuración con nombres cfg1, cfg2, cfg3
- un cliente mongos en el puerto por defecto (27017)
- 2 shards (particiones), a y b, cada una un conjunto réplica formado por 3 servidores

Los nombres y puertos vienen indicados en este gráfico:



1) Crear:

a) Las carpetas de datos necesarias (desde el terminal). Consejo: crearlas dentro de una carpeta (por ejemplo “practshard”). Copiar en la solución el comando para crear las carpetas de datos (no la carpeta exterior “practshard”)

b) Crear los servidores de configuración. Copiar las instrucciones que permiten hacerlo a la solución.

c) Iniciar los servidores de datos que formarán las particiones. No olvidar poner el nombre del conjunto réplica (a o b). Copiar las instrucciones en la solución

d) Crear un servicio cliente mongos. Poner como nombre del fichero log (opción logpath) mongoslog.txt
Recordar que aquí se usan los nombres completos de los servidores de configuración. Estos nombres serán de la forma nombrehost.local:puerto. Para averiguar el nombre del host en un terminal de linux se puede teclear “hostname”.

Para segurarnos de que todos los servidores y servicios se han iniciado, podemos teclear en el terminal

ps -A | grep “mongo”

deben aparecernos 10 líneas: 9 para servidores (mongod) y 1 para el cliente mongos

Solución

a) Comando(s) para crear las carpetas desde el terminal:
mkdir a1 a2 a3 b1 b2 b3 cfg1 cfg2 cfg3

b) Creación de servidores de configuración

```
mongod --configsvr --dbpath cfg1 --port 26050 --fork --logpath cfg1/log --logappend
mongod --configsvr --dbpath cfg2 --port 26051 --fork --logpath cfg2/log --logappend
mongod --configsvr --dbpath cfg3 --port 26052 --fork --logpath cfg3/log --logappend
```

c) Creación de servidores de datos

```
mongod --shardsvr --replSet a --dbpath a1 --logpath a1/log --port 27101 --fork
--logappend --oplogSize 50
mongod --shardsvr --replSet a --dbpath a2 --logpath a2/log --port 27102 --fork
--logappend --oplogSize 50
mongod --shardsvr --replSet a --dbpath a3 --logpath a3/log --port 27103 --fork
--logappend --oplogSize 50
mongod --shardsvr --replSet b --dbpath b1 --logpath b1/log --port 27201 --fork
--logappend --oplogSize 50
mongod --shardsvr --replSet b --dbpath b2 --logpath b2/log --port 27202 --fork
--logappend --oplogSize 50
mongod --shardsvr --replSet b --dbpath b3 --logpath b3/log --port 27203 --fork
--logappend --oplogSize 50
```

d) Creación del servicio cliente mongos:

```
mongos --configdb puck.local:26050,puck.local:26051,puck.local:26052 --fork
--logappend --logpath mongoslog.txt
```

2) Activar los dos conjuntos réplica. Recordar que conviene hacerlo desde un cliente conectado a un miembro de la réplica. Copiar en la solución las instrucciones que se han utilizado dentro del cliente mongo así como las de sistema operativo para conectar con mongo. Escribirlas en secuencia. No hace falta indicar si la instrucción es en el terminal o dentro de mongo.

Aviso: revisar bien los “corta-pega”; los carga el diablo.

Solución

```
mongo –port 27101
config = {_id: "a",
          members: [ {_id:0, host:"puck.local:27101" },{_id:1,
host:"puck.local:27102" },{_id:2, host:"puck.local:27103" },   ] }
rs.initiate(config)
```

```
mongo –port 27201
config = {_id: "b",
          members: [ {_id:0, host:"puck.local:27201" },{_id:1,
host:"puck.local:27202" },{_id:2, host:"puck.local:27203" },   ] }
rs.initiate(config)
```

3) Crear las dos particiones

Copiar en la solucion las instrucciones que se han utilizado dentro del cliente mongo así como las de sistema operativo para conectar con mongo. Escribirlas en secuencia. No hace falta indicar si la instrucción es en el terminal o dentro de mongo.

Solución

```
mongo
sh.addShard("a/puck.local:27101")
{ "shardAdded" : "a", "ok" : 1 }
mongos> sh.addShard("b/puck.local:27201")
{ "shardAdded" : "b", "ok" : 1 }
```

4) Conectar con mongo (prompt mongos).
Escribir

```
use pin-pon
for (i=0;i<1000;i++){db.cole.insert({"pin":i, "pon":1000-i})}
sh.status()
```

¿Está particionada la base de datos pin-pon? ¿Qué parte de la respuesta que muestra sh.status lo indica?

Solución.

¿Está particionada la base de datos pin-pon? (Sí/No): No
¿Qué parte de la respuesta que muestra sh.status lo indica? (una frase) "partitioned":false en el documento para la base de datos.

5) Seguimos en mongo (prompt mongos) y usando la base de datos pin-pon. Ahora tecleamos:

```
db.cole.stats()
```

¿Está particionada la colección cole? ¿Qué parte de la respuesta de db.cole.stats() lo indica?

Solución
¿Está particionada la colección cole? (sí/no): no
¿Qué parte de la respuesta de db.cole.stats() lo indica? "sharded: false"

6) Teclear en mongo (mongos):

```
use aleatorio
db.dados.drop()
```

La colección dados representa una simulación de muchas tiradas de un dado. Los documentos, que insertaremos luego, son de la forma

```
{tirada:0, dado: 5, acierto:"huy...casi adivinas el resultado"}
```

con "tirada" un valor único que indica el orden en la simulación, dado el valor que ha salido, y *acierto* un mensaje que indica si se ha adivinado el resultado de la tirada o no.

Haz que la colección *dados* admita “sharding”. Al hacerlo selecciona la clave “dado”

Copia en la solución las instrucciones que han permido que la colección admita particiones.

Solución

```
sh.enableSharding("aleatorio")
db.dados.createIndex({dado:1},{unique:false})
sh.shardCollection("aleatorio.dados", {dado:1},false)
```

7) Ahora ejecutamos (tardará un rato):

```
for (i=1;i<=20000;i++){tira= Math.floor(Math.random()*6)+1;
                       apuesta=Math.floor(Math.random()*6)+1;
                       db.dados.insert({tirada:i,
                                        dado:tira,
                                        acierto: (tira==apuesta ? "Has acertado" :
                                       "Bertoldo, has perdido. Ten en cuenta que las vidas de la mayoría de la gente están dirigidas por el deseo y el miedo. El deseo es la necesidad de añadirte algo para poder ser tú mismo más plenamente. Todo miedo es el miedo de perder algo y, por tanto, de sentirte reducido y de ser menos de lo que eres. Estos dos movimientos oscurecen el hecho de que el Ser no puede ser dado ni quitado. El Ser ya está en ti en toda su plenitud, ahora (Eckhart Tolle)" )}); }
s = db.dados.stats()
```

¿Cuántos documentos tiene cada shard? Obtener esta información a partir de la variable s

Pista1: Investiga el formato del documento almacenado en s y utiliza la notación . :
s.XXXX.YYYY…...

Indica en la solución, la cantidad de documentos por particion el comando que has utilizado para obtener la información y las expresiones a partir de s que permiten obtener esta información

Solución:

Número de documentos en la partición a: 6566
Número de documentos en la partición b: 13434
Nota: estos valores pueden variar, pero deben verificar que a+b = 20000
Expresiones:
- s.shards.a.count
- s.shards.b.count

8) Ahora vamos a ver cómo afecta la existencia de las particiones a los planes de ejecución, y por tanto a la eficiencia.

```
exp = db.dados.explain("executionStats")
exp.find({tirada:{'$gte':9980}},{_id:0,tirada:1})
```

a) ¿Cuántos shards se han consultado? (consultar el array shards)
b) ¿Cuántas claves?
c) ¿Cuántos documentos?

Ayuda: el documento que devuelve puede ser muy largo, mejor copiarlo y verlo en un editor (por ejemplo gedit)

Solución
a) ¿Cuántos shards se han consultado? 2
b) ¿Cuántas claves? 0
c) ¿Cuántos documentos? 20000

9) Análogo pero para la clave *dado*:

```
exp = db.dados.explain("executionStats")
exp.find({dado:5},{_id:0,dado:1})
```

a) ¿Cuántos shards se han consultado?
b) ¿Cuántas claves?
c) ¿Cuántos documentos?

Solución
a) ¿Cuántos shards se han consultado? 1
b) ¿Cuántas claves? 3281
c) ¿Cuántos documentos? 0

10) Teclear: use config
Escribir una consulta que indique el shard del valor dado:4 . Copiarla en la solución.
Pista: db.chunks.find(...)

Solución
db.chunks.find({ns:"aleatorio.dados", "min.dado":{"$lte":4}, "max.dado":{"$gte":4}},{shard:1,_id:0})