Actividad 5: Conceptos y Comandos básicos del particionamiento en bases de datos NoSQL

Brayan Steven Bonilla Castellanos

Juan Carlos Monsalve Gómez

Corporación Universitaria Iberoamericana

Ingeniería de Software

Bases de datos avanzadas

Requerimientos no funcionales

- La base de datos debe permitir realizar todas las operaciones de consulta necesarias.
- La base de datos debe estar disponible 24/7
- Se debe garantizar la seguridad de la información.
- Debe permitir el acceso de varios usuarios al mismo tiempo (concurrencia)
- Se debe garantizar la fiabilidad de la información.
- Debe permitir la escalabilidad horizontal para el crecimiento del negocio.

Enlace Repositorio GIT

https://github.com/jcmonsalveg/Actividad-5---ParticionamientoMongo

Enlace Video

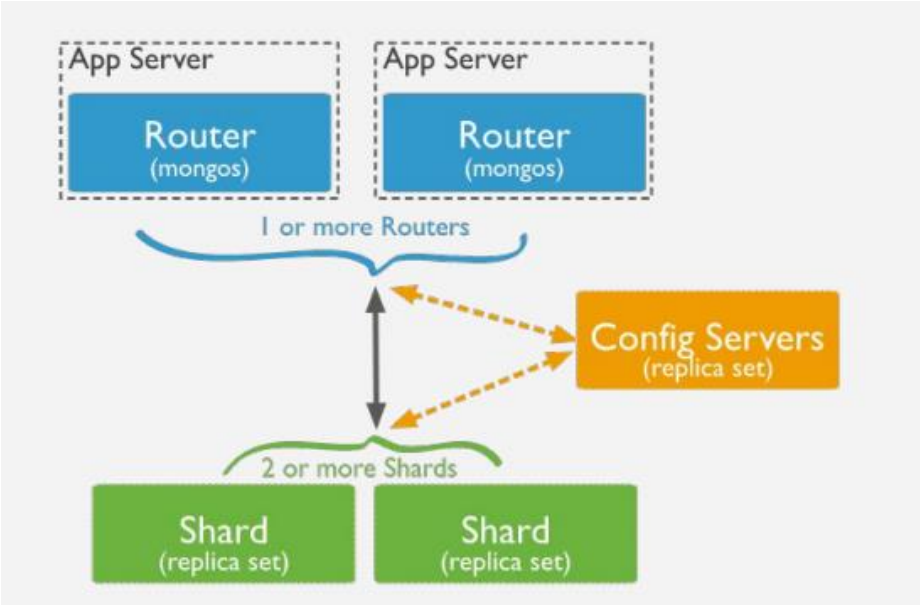
https://youtu.be/wmj3AWisOeo

Sharding

Fragmento: cada fragmento contiene un subconjunto de los datos fragmentados. Cada fragmento se puede implementar como un conjunto de réplicas.

● mongos : mongos actúa como un enrutador de consultas, proporcionando una interfaz entre las aplicaciones cliente y el clúster fragmentado. A partir de MongoDB 4.4, mongos puede admitir lecturas de cobertura para minimizar las latencias.

● Servidores de configuración: los servidores de configuración almacenan metadatos y ajustes de configuración para el clúster.



Sharding MongoDB

Generar el Replica Set de los Config Server

Se generan los 3 ficheros de configuración de los Config Servers

Puertos: 28017 28117 28217

cfgsvr1.cfg

```
net:
    bindIp: 127.0.0.1
    port: 28017
sharding:
    clusterRole: configsvr
replication:
    replSetName: "replicaConfSvr"
storage:
    dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/cfgsvr1/"
systemLog:
    destination: file
    path: "C:/Program Files/MongoDB/Server/3.4/log/cfgsvr1.log"
    logAppend: true
```

cfgsvr2.cfg

```
net:
    bindIp: 127.0.0.1
    port: 28117
sharding:
    clusterRole: configsvr
replication:
    replSetName: "replicaConfSvr"
storage:
    dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/cfgsvr2/"
systemLog:
    destination: file
    path: "C:/Program Files/MongoDB/Server/3.4/log/cfgsvr2.log"
    logAppend: true
```

cfgsvr3.cfg

```
net:
    bindIp: 127.0.0.1
    port: 28217
sharding:
    clusterRole: configsvr
replication:
    replSetName: "replicaConfSvr"
storage:
    dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/cfgsvr3/"
systemLog:
    destination: file
    path: "C:/Program Files/MongoDB/Server/3.4/log/cfgsvr3.log"
    logAppend: true
```

Se inician los 3 Config Servers

```
mongod --config cfgsvr1.cfg --install --serviceName
mongod --config "C:\Program Files\MongoDB\Server\3.4\etc\cfgsvr1.cfg" --install --serviceName Mongocfgsvr1
--serviceDisplayName "MongoDB Config Server 1"
mongod --config "C:\Program Files\MongoDB\Server\3.4\etc\cfgsvr2.cfg" --install --serviceName Mongocfgsvr2
--serviceDisplayName "MongoDB Config Server 2"
mongod --config "C:\Program Files\MongoDB\Server\3.4\etc\cfgsvr3.cfg" --install --serviceName Mongocfgsvr3
--serviceDisplayName "MongoDB Config Server 3"
```

Conectarse a la interface localhost de uno de los config servers para inicializar el replica set de los config servers

```
rs.initiate(
  {
    _id: "replicaConfSvr",
    configsvr: true,
    members: [
      { _id : 0, host : "localhost:28017" },
      { _id : 1, host : "localhost:28117" },
      { _id : 2, host : "localhost:28217" }
    ]
  }
)
```

Generar el Primer Shard Replica Set

Se generan los ficheros de configuración uno de los parametros importantes es el sharding.clusterRole

Puertos: 29017 29117 29217

shard1pri.cfg

```
net:
   bindIp: 127.0.0.1
   port: 29017
sharding:
   clusterRole: shardsvr
replication:
   replSetName: "replicaShard1"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/shard1pri/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/shard1pri.log"
   logAppend: true
```

shard1sec.cfg

```
net:
   bindIp: 127.0.0.1
   port: 29117
sharding:
   clusterRole: shardsvr
replication:
   replSetName: "replicaShard1"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/shard1sec/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/shard1sec.log"
   logAppend: true
```

shard1arb.cfg

```
net:
   bindIp: 127.0.0.1
   port: 29217
sharding:
   clusterRole: shardsvr
replication:
   replSetName: "replicaShard1"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/shard1arb/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/shard1arb.log"
   logAppend: true
```

Se inician los 3 Servers del Shard1

```
mongod --config "C:\Program Files\MongoDB\Server\3.4\etc\shard1pri.cfg" --install --serviceName Mongosh1pri
--serviceDisplayName "MongoDB Shard 1 Primary"
mongod --config "C:\Program Files\MongoDB\Server\3.4\etc\shard1sec.cfg" --install --serviceName Mongosh1sec
--serviceDisplayName "MongoDB Shard 1 Secondary"
mongod --config "C:\Program Files\MongoDB\Server\3.4\etc\shard1arb.cfg" --install --serviceName Mongosh1arb
--serviceDisplayName "MongoDB Shard 1 Arbiter"
```

Conectarse a la interface localhost de uno de los servers del shard1 para inicializar el replica set de este shard.

Con esta configuración he querido forzar cual es el primario por defecto, cual el secundario y cual es árbitro.

```
rs.initiate(
  {
    _id: "replicaShard1",
    members: [
      { _id : 0, host : "localhost:29017", priority : 20 },
      { _id : 1, host : "localhost:29117", priority : 10 },
      { _id : 2, host : "localhost:29217", arbiterOnly : true }
    ]
  }
)
```

Generar el Segundo Shard Replica Set

Se generan los ficheros de configuración uno de los parametros importantes es el sharding.clusterRole

Puertos: 29317 29417 29517

shard2pri.cfg

```
net:
   bindIp: 127.0.0.1
   port: 29317
sharding:
   clusterRole: shardsvr
replication:
   replSetName: "replicaShard2"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/shard2pri/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/shard2pri.log"
   logAppend: true
```

shard2sec.cfg

```
net:
   bindIp: 127.0.0.1
   port: 29417
sharding:
   clusterRole: shardsvr
replication:
   replSetName: "replicaShard2"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/shard2sec/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/shard2sec.log"
   logAppend: true
```

shard2arb.cfg

```
net:
   bindIp: 127.0.0.1
   port: 29517
sharding:
   clusterRole: shardsvr
replication:
   replSetName: "replicaShard2"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/shard2arb/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/shard2arb.log"
   logAppend: true
```

Se inician los 3 Servers del Shard2

```
mongod --config shard2pri.cfg
mongod --config shard2sec.cfg
mongod --config shard2arb.cfg
```

Conectarse a la interface localhost de uno de los servers del shard2 para inicializar el replica set de este shard.

Con esta configuración he querido forzar cual es el primario por defecto, cual el secundario y cual es árbitro.

```
rs.initiate(
  {
    _id: "replicaShard2",
    members: [
      { _id : 0, host : "localhost:29317", priority : 20 },
      { _id : 1, host : "localhost:29417", priority : 10 },
      { _id : 2, host : "localhost:29517", arbiterOnly : true }
    ]
  }
)
```

Configurar los routers

Generar el fichero de configuración del router

router1.cfg

```
net:
   bindIp: 127.0.0.1
   port: 27017
sharding:
   configDB: replicaConfSvr/localhost:28017,localhost:28117,localhost:28217
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/router1.log"
   logAppend: true
```

Arrancar el mongos

```
mongos --config router1.cfg
```

Conectarse a la interface localhost del router

```
admin = db.getSiblingDB("admin")
admin.createUser(
  {
    user: "admin",
    pwd: "Password1",
    roles: [ { role: "userAdminAnyDatabase", db: "admin" }, { role: "clusterAdmin", db: "admin" }, { role:
    "readAnyDatabase", db: "admin" }, { role: "readWriteAnyDatabase", db: "admin" }, { role:
    "userAdminAnyDatabase", db: "admin" } ]
  }
)
```

Autenticarse

```
db.getSiblingDB("admin").auth("admin", "Password1" )
```

Como alternativa, conecte un nuevo mongo shell al miembro del conjunto de réplicas de destino mediante los parámetros -u <username>, -p <password> y --authenticationDatabase "admin". Debe usar la excepción Localhost para conectarse a los mongos.

```
mongo -u "fred" -p "changeme1" --authenticationDatabase "admin"
```

Añadir fragmentos al clúster

Para continuar, debe estar conectado a mongos y autenticado como usuario administrador del clúster para el clúster fragmentado.

```
sh.addShard( "replicaShard1/localhost:29017")
```

Habilitar fragmentación para una base de datos

Para continuar, debe estar conectado a mongos y autenticado como usuario administrador del clúster para el clúster fragmentado.

Este es el administrador del clúster para el clúster fragmentado y no el administrador del clúster local del fragmento.

Habilitar la fragmentación en una base de datos hace posible fragmentar colecciones dentro de la base de datos. Utilice el método sh.enableSharding() para habilitar la fragmentación en la base de datos de destino.

```
sh.enableSharding("torneo_deportivo")
```

Fragmentar una colección

Para continuar, debe estar conectado a mongos y autenticado como usuario administrador del clúster para el clúster fragmentado.

Si la colección ya contiene datos, debe crear un índice en la clave de fragmento usando el método db.collection.createIndex() antes de usar shardCollection().

Si la colección está vacía, MongoDB crea el índice como parte de sh.shardCollection().

```
sh.shardCollection("torneo_deportivo.deportistas",{"nombre":"Juan"})
```