MongoDB – Sharding, Réplication

Objectif
Découvrir le partitionnement de données dans MongoDB

Sharding
On va découvrir le modèle de partitionnement de données dans MongoDB. Le mécanisme de ce SGBD utilise une fragmentation horizontale où les documents d'une collection sont disposés sur des machines différentes suivant les valeurs d'une clé de sharding.
Ceci est supporté par l'exécutable **mongos** dans le répertoire bin/ de l'installation de MongoDB. Mongos gère la distribution des données et le routage des requêtes. Lorsque les données qui satisfont une requête se trouvent sur plusieurs machines, mongos gère la fusion des résultats intermédiaires.

MongoDB utilise le concept de **Config Server** pour stocker les métadonnées d'un cluster de shards. Ces métadonnées reflètent l'état et l'organisation de toutes les données et de tous les composants d'un cluster de shard. Les métadonnées stockent la liste des morceaux (chunks) de chaque shard et les intervalles qui définissent les chunks. Les config servers gèrent également l'authentification, la distribution des verrous (pas étudié dans ce TP).
Chaque shard cluster doit avoir son config server et il ne faut pas utiliser le même config server pour plusieurs clusters.

A partir de la version 3.2, un config server doit être déployé comme un replica set (cf tp sur réplication).

1. Comme pour le TP de réplication avec MongoDB, il faut créer plusieurs répertoires. On va créer 1 répertoire dbs dans /tmp puis 4 répertoires dans ce dernier : config1, config2, shard31 et shard32. Chacun de ces répertoires contiendra un répertoire data. Vous devez donc avoir une organisation de ce type

```
/tmp/dbs
        config1
                data
        config2
                data
        shard31
                data
        shard32
                data
```

2. on va utiliser 5 ports et plusieurs daemons de mongoDB :
shard controller sur le port 27017
Config server sur les ports 27019 et 27020 associés respectivement aux répertoires config1 et config2
shard31 et shard32 sur respectivement 27031 et 27032, associés respectivement avec shard31/data et shard32/data

Vous allez lancer 4 consoles sur votre installation mongodb

3.
Dans une première console dédié à un config serveur, saisir la commande suivante :
```
bin/mongod --configsvr --replSet configReplSet --port 27019 --dbpath
/tmp/dbs/config1
```

Dans la seconde console dédié à un config serveur, saisir :
```
bin/mongod --configsvr --replSet configReplSet --port 27020 --dbpath
/tmp/dbs/config2
```

Dans la console de 27019, ouvrir un nouvel onglet, lancer mongo et initialiser le replica set (cf tp#1, celle-ci doit indiquer que le replica set comporte les serveurs de 27019 et 27020). Vérifier l'état des 2 replica sets du config server du shard cluster

4.

On s'occupe maintenant des 2 shards en interagissant sur les 2 autres consoles.
Dans la première, saisir la commande suivante :

```
bin/mongod --shardsvr --replSet sh0 --port 27031 --dbpath
/tmp/dbs/shard31
```

Dans la seconde

```
bin/mongod --shardsvr --replSet sh1 --port 27032 --dbpath
/tmp/dbs/shard32
```

Il faut maintenant initialiser les 2 replica sets (sh0 et sh1).
On peut lancer la configuration du replica set directement depuis le lancement de mongo avec : `--eval "rs.initiate()"`

5.

Dans une nouvelle console, on lance le contrôleur de shard avec

```
bin/mongos --configdb configReplSet/localhost:27019,localhost:27020 --port
27017
bin/mongo --port 27017
```

Observez l'invitation de la console !
On reste dans cette console.
Le mode sharding de mongoDB partitionne les données par chunks (morceaux) de données. Par défaut, la taille d'un chunk est de 64Mo. On ne va pas créer une grosse base de données donc on va limiter la taille d'un chunk. Pour cela, on se connect à la base config :

```
use config
```

puis, on saisit l'instruction suivante :

```
db.settings.save( { _id:"chunksize", value: 1 } )
```

où 1 indique la taille en Mo d'un chunk.

On se connecte maintenant à une nouvelle base de données :

```
use testdb
```

On va spécifier les serveurs de shard au controller de shards

```
sh.addShard( "sh0/localhost:27031");
sh.addShard( "sh1/localhost:27032");
```

On peut voir le status du sharding avec : db.printShardingStatus()

6.

On va spécifier la base de données et la collection à partitionner avec les commandes suivantes :
sh.enableSharding("testdb")
sh.shardCollection("testdb.col1", {testkey : 1})

Le champ testkey est donc la clé de sharding de la collection « col1 » de la base de données « testdb ».
Toujours depuis le shard contrôleur, insérer 10.000 documents dans la collection col1 où les valeurs de testkey prises aléatoirement dans un intervalle [0,1000] (adapter le code javascript du tp sur la réplication) et text prendra la même valeur textuelle (d'une centaine de caractères) dans tous les documents de la collection.

Utiliser les commandes `db.collection.getShardDistribution()` et `db.adminCommand({getShardMap:1})`
Vérifier le nombre d'enregistrements dans la collection dans le shard controleur, le shard0 et le shard1.

NB : count() est utilisé dans getShardDistribution() mais donne une estimattion. Avec countDocuments({}), on obtient un résultat exact

7. Ouvrez les shells pour les ports 27031 et 27032. Vérifier que la base de données testdb contient bien la collection col1. Vérifier son état dans les 2 consoles (avec un count sur la collection). Vérifier avec l'état indiqué depuis 27017.