MongoDB - Cluster

ReplicaSet & Sharding

Travaux Pratiques

ESILV
nicolas.travers (at) devinci.fr

Chapitre

Table des matières

Chapitre 1
tructions pour la gestion d'un cluster MongoDB

Pour l'ensemble des travaux pratiques sur le cluster, il faudra utiliser les VMs à disposition sur : `https://stargate.exaduo.fr/`.

— Lancer les deux machines virtuelles sur compte que nous appelerons ici : **VM1** et **VM2** ;
— Ouvrir deux consoles en `SSH` sur les VMs que nous appelerons : **C1** et **C2**
— Console *mongo* avec nom du serveur : `mongo --host XXXX`

1.1 Lancement d'un serveur avec fichier de configuration

Pour que le lancement de l'instruction *mongod* soit plus simple à chaque serveur, vous allez pouvez utiliser un fichier de configuration comme suit :

— Le répertoire "conf" contient les fichiers de configurations :
  — `conf/mongo_RS1.conf` → config avec ReplicaSet *RS1*
  — `conf/mongo_RS2.conf` → config avec ReplicaSet *RS2*
  — `conf/mongo_configSvr.conf` → config pour le lancement d'un configServer
  — `conf/mongos.conf` → config pour le routeur '*mongos*'
— Pour lancer le serveur avec le fichier de configuration dans une console SSH (C1 ou C2) :

```
mongod -f conf/mongo_RS1.conf
```

— Vous gardez la main dans la console (option 'fork' du fichier de config)
— Pour vérifier que le processus est en cours :

```
ps -aux | grep mongo
```

— Vous pouvez consulter le contenu du fichier de configuration avec *vim* (sous-section suivante).

1.2 Éditeur de texte en ligne de commande : *vim*

Pour éditer vos fichiers, vous pouvez utiliser *vim*. Prenons l'exemple du fichier de configuration :

```
vim conf/mongo_RS1.conf
```

Voici quelques commandes à connaître :

— `i` → mode édition (insert)
— `escape` → quitter le mode en cours
— `:w` → enregistrer (write)
— `:q` → quitter (quit)

Autres instructions : `http://www.yolinux.com/TUTORIALS/LinuxTutorialAdvanced_vi.html`

1.3 Édition du fichier de configuration

Voici la signification des différentes lignes du fichier de configuration :

bind_ip=mongodbXXX	**Nom du serveur**. À utiliser dans la partie *replicaSet* et *sharding*
port=27017	Port utilisé pour la connexion
logpath=....	Localisation du fichier de log
fork=true	Ce qui permet de récupérer la main après lancement
dbpath=....	Localisation des fichiers de la base de données
replSet=RS1	Nom du **ReplicaSet**. À changer si le nom doit être différent
shardsvr=true	Est-ce que le serveur est ouvert au **sharding**

1.4 Éteindre MongoDB dans une console MongoDB

— Lancer la console **SSH** ;

— Éteindre le serveur :

```
mongod -f conf/mongo_RS1.conf -shutdown
```

— C'est fait. Vous pouvez quitter la console mongo.

1.5 Lancement automatique

Si vous voulez aller plus loin, vous pouvez faire démarrer le serveur automatiquement avec le fichier de configuration. Pour cela, il faudra créer un service `init.d`.

`https://github.com/mongodb/mongo/blob/master/rpm/init.d-mongod`

Il faudra changer quelques valeurs pour l'adapter à votre VM. Si vous y arrivez, envoyez-moi votre fichier de config et instructions pour lancement pour que je puisse valider.

Cela pourra éventuellement donner des points bonus. Qui sait ? À croire que mettre une telle instruction à la fin d'un chapitre que personne ne va lire est fait exprès...

Chapitre 2

Tolérance aux pannes

Nous voulons ici tester le système de tolérance aux pannes de MongoDB. Pour cela, nous allons lancer des serveurs avec des `ReplicaSets`.

2.1 Installation d'un replicaSet

Nous allons lancer un replicatSet avec 3 ou 4 serveurs. Pour les instructions de base, se référer au chapitre 1.

Par groupe de 3 ou 4 élèves :

2.1.1 Chaque élève lance un serveur avec le fichier de configuration `mongo_RS1.conf` dans la console **C1**

2.1.2 Le *master* va lancer le replicaSet dans la console **mongo** :

```
rs.initiate ();
```

2.1.3 Il récupère le **nom de serveur** de chaque membre de son groupe (nom sur stargate.edunao.fr ou dans le fichier de configuration à la ligne 'bind_ip').

2.1.4 Insertion d'un serveur au replicatSet (remplacer XXXX par le nom du serveur) :

```
rs.add('XXXXXXXXX:27017')
```

2.1.5 Chaque élève peut vérifier le statut du replicaSet (s'il est PRIMARY ou SECONDARY) :

```
rs.status();
```

2.1.6 Vous pouvez également rajouter un arbitre :

```
rs.addArb("XXXXX:27017");
```

2.2 Test de réplication

2.2.1 Se connecter au replicatSet (Studio3t, Robo3t, mongo, etc.) en précisant la connexion au replicaSet RS1 (sur votre VM).

2.2.2 Lancer la requête suivante sur la collection `tourPedia.paris`

```
db.paris.count();
```

2.2.3 Éteindre le serveur PRIMARY (section 1.4)

2.2.4 Vérifier qui devient le PRIMARY

2.2.5 Relancer la requête

2.2.6 Relancer le serveur éteint. Que se passe t-il ?

2.2.7 Tester l'insertion d'un document et vérifier son existence (après extinction du serveur)

2.3 Configuration rapide

Il est possible de rajouter une liste de serveur lors de l'initialisation :

```
rsconf = {
            _id: "RS1",
            members: [
                        {_id: 0,host: "XXX:27017"},
                        {_id: 1,host: "YYY:27017"},
                        {_id: 2,host: "ZZZ:27017",arbiterOnly:true},
                      ]
          };
rs.initiate(rsconf);
```

Chapitre 3

Création d'un cluster

Nous voulons ici tester le système de distribution avec le *sharding* de MongoDB.

3.1 Installation du cluster

Nous allons lancer un cluster avec 3 shards. Pour les instructions de base, se référer au chapitre 1.

Se mettre par groupe de 3 ou 4 élèves

3.1.1 Définition de l'architecture. Par groupe, définir parmi toutes les VM disponibles celles qui vont jouer les rôles suivants :

CS1 **ConfigServer** : Un replicaSet pour gérer l'ensemble du cluster (1 serveur minimum, normalement 3)

MS **mongos** : Un routeur qui va recevoir les requêtes.
Normalement 3 routeurs. On peut en mettre 1 par élève pour faciliter les connexions extérieures)

Sx **shards** : Le reste peut être utilisé comme shards (replicatSet de 1 serveur)

3.1.2 Pour **CS1** :

(a) Lancer **CS1**. Utiliser le fichier de config "`mongo_configSvr.conf`"

(b) Lancer le replicatSet

```
mongo --host XXXX --port 27018
rs.initiate()
```

3.1.3 Pour les shards **Sx** :

(a) 1 shard = 1 replicaSet
⚠ changer le nom du RSx dans le fichier de configuration RS1/RS2 (ou générer un nouveau)

(b) Lancer **Sx**. Utiliser le fichier de config "`mongo_RSx.conf`" (remplacer le x par le numéro du replicaSet)

(c) La première fois, initialiser le replicaSet (section 2.1)

3.1.4 Pour le routeur **MS** :

(a) Changer le fichier de config "`mongos.conf`" pour cibler **CS1** : `configdb=configSvr/XXXXXX:27018`

(b) Lancer le **MS** avec le fichier de config

(c) Se connecter via console SSH au routeur et lancer la console mongo :

```
mongo --host XXXXX --port 30000
```

(d) Ajouter les *shards* :

```
sh.addShard("RS1/XXX:27017")
sh.addShard("RS2/YYY:27017")
sh.addShard("RS3/ZZZ:27017")
```

3.2 Initialisation du sharding

3.2.1 Dans la console mongo sur **MS** :

```
use test;
sh.enableSharding("test");
db.createCollection('testSharding');
sh.shardCollection('test.testSharding', {"_id": 1});
```

3.2.2 Importer une collection en utilisant les paramètres suivants :

Serveur	Port	Database	Collection
MS (nom de votre configServer)	30000	test	testSharding

3.2.3 Dans la console mongo de **MS**, vérifier le statut du cluster :

```
sh.status();
```

Vous pourrez constater la prise en contact des shards (et leur replicaSet), des configServer, et de l'emplacement des chunks de votre collection.

3.3 Inspecter la configuration

— Connectez-vous à **MS** (Studio3t, Robo3t, mongo, etc.)
— Compter le nombre de documents présents dans `test.testSharding`
— Connectez-vous à **Sx** et faites de même. Pourquoi le résultat peut être différent ?
— Tester deux requêtes différentes :
  — Recherche de la clé `{_id:1}`
  — Recherche d'une clé `{XXX:YYY}`

  Avec la fonction '.explain()', constater l'utilisation du *sharding*.
— Reprendre le processus de la section 3.2, mais en faisant un *sharding* sur la clé XXX dans une nouvelle collection.

3.4 Requêtes

Vous avez déjà fini ? Vous pouvez testé les requêtes *mapReduce* ou *aggregate* sur la collection *tourPedia.paris*