Rapport TP3

Partitionnement

Addou Aicha Amira

Bombay Marc

3ème Année Ingénieur - INFO - Apprentissage

Introduction

MongoDB, en tant que base NoSQL orientée documents, propose le sharding comme mécanisme de passage à l'échelle horizontale.

Ce TP a pour objectif de mettre en œuvre une architecture MongoDB shardée et d'étudier le fonctionnement du partitionnement des données.Dans ce TP, on met en place un cluster shardé en local composé de :

- 1 replica set pour les config servers : replicaconfig (un seul nœud pour le TP)
- 2 shards : replicashard1 et replicashard2
- 1 routeur mongos

Partie 1 - Mise en place du sharding

1. Connexion et création des repertoires

Une fois connecté, nous allons dans un premier temps créer des répertoire de stockage. Un pour le répertoire et deux autres pour les shards.

```
mkdir -p configsvrdb
```

```
mkdir -p serv1
```

```
mkdir -p serv2
```

2. Démarrer le Config Server

Terminal configsvr :

```
mongod --configsvr --replSet replicaconfig --dbpath configsvrdb --port 27019
```

Initialisation du replica set (terminal client-init) :

```
mongosh --port 27019
```

Dans mongosh :

```
rs.initiate({
  _id: "replicaconfig",
  members: [{ _id: 0, host: "localhost:27019" }]
})
```

Vérification :

```
rs.status()
```

3. Démarrer le routeur mongos

Lancer `mongos` sur le port 27017 (port MongoDB “standard”) :

```
mongos --configdb replicaconfig/localhost:27019 --port 27017
```

4. Démarrer les deux shards

Terminal shard1 :

```
mongod --replSet replicashard1 --dbpath serv1 --shardsvr --port 20004
```

Terminal shard2 :

```
mongod --replSet replicashard2 --dbpath serv2 --shardsvr --port 20005
```

Initialiser chaque replica set :

Shard1 :

```
mongosh --port 20004
rs.initiate({
  _id: "replicashard1",
  members: [{ _id: 0, host: "localhost:27018" }]
})
```

Shard2 :

```
mongosh --port 20005

rs.initiate({

  _id: "replicashard2",

  members: [{ _id: 0, host: "localhost:27019" }]

})
```

5. Connecter et ajouter les shards

Connexion à mongos dans le terminal Client :

```
mongosh --port 27017
```

Ajouter les shards

```
sh.addShard("replicashard1/localhost:20004")

sh.addShard("replicashard2/localhost:20005")
```

Vérification :

```
sh.status()
```

6. Activer le sharding sur une collection

On utilise la base et collection du sujet :

Base : `lesfilms`

Collection : `films`

Dans mongosh connecté à mongos on active le sharding :

```
sh.enableSharding("lesfilms")
```

Sharder la collection, Shard de la collection (clé : titre) :

```
sh.shardCollection("Films.films", { "titre": 1 })
```

création index :

```
use lesfilms

db.films.createIndex({ titre: 1 })
```

7. Insertion des données (script Python)

Le script `monappunparun.py` insère des films générés aléatoirement, document par document.

Points importants :

- DB : `mabasefilms`
- Collection : `films`
- Connexion : `MongoClient("mongodb://localhost:27017/")`

Donc, si tu lances `mongos` sur 27017, tu peux exécuter le script sans modification en lancant :

```
python3 monappunparun.py
```

8. Observation / vérification du sharding

Vérifier l'état du cluster:

```
sh.status()
```

Vérifier la distribution des données :

```
use mabasefilms

db.films.getShardDistribution()
```

Voir les chunks dans la base config :

```
use config

db.chunks.find({ ns: "mabasefilms.films" }).pretty()
```

Surveiller le balancer :

```
sh.getBalancerState()
```

```
sh.isBalancerRunning()
```

9. Analyse

- Au début, la collection shardée possède un chunk initial sur un shard.
- Quand la taille du chunk dépasse un seuil, MongoDB effectue un split (création de 2 chunks plus petits).
- Le balancer surveille l'équilibre entre shards (chunks) et peut déclencher des migrations de chunks vers l'autre shard.
- Avec une shard key mal choisie (ex. monotone), on peut créer un hot shard (un shard reçoit presque toutes les écritures).

Partie 2 - Réponses aux questions

1. Qu'est-ce que le *sharding* dans MongoDB et pourquoi est-il utilisé ?

Il s'agit d'un partitionnement horizontal. Une collection est découpée et répartie sur plusieurs *shards*.

Il est utilisé pour gérer de très gros volumes de données, pour améliorer les performances de lecture/écriture.

2. Quelle est la différence entre le sharding et la réplication dans MongoDB ?

Le sharding gère la **répartition des données** entre plusieurs serveurs (*shards*).

La réplication **copie les données**. Elle sert pour la tolérance aux pannes.

On note que l'on utilise souvent les deux ensemble.

3. Quels sont les composants d'une architecture shardée (mongos, config servers, shards) ?

Pour MongoDB:

- `mongos` : Routeur
- *Config Server* : Contient les métadonnées (expliquées plus bas)
- *Shards* : Là où sont stockées les données. Peuvent être des *Replica Set* (cf. réplication).

4. Quelles sont les responsabilités des config servers (CSRS) dans un cluster shardé ?

Les config servers ont plusieurs responsabilités.

Ils doivent maintenir la *clé de sharding* (définie en 7), la répartition des chunks. Ils doivent aussi coordonner le balancer (défini en 11).

5. Quel est le rôle du mongos router ?

Le routeur (`mongos` sous MongoDB) constitue le point d'entrée pour les clients. Après analyse des requêtes, il les redirige vers les shards concernés et agrège ensuite les résultats si nécessaire.

6. Comment MongoDB décide-t-il sur quel shard stocker un document ?

La décision est faite basée sur la *clé de sharding*. MongoDB calcule sa valeur, trouve le *chunk* qui lui correspond et stocke en conséquent sur le *shard* qui possède ce *chunk*.

Les *chunks* sont définis plus loin (9).

7. Qu'est-ce qu'une clé de sharding et pourquoi est-elle essentielle ?

Il s'agit d'un ou plusieurs champs, utilisés pour déterminer comment les documents doivent être distribués. La clé sert aussi à router les requêtes efficacement si le filtre est fait sur les mêmes champs que la clé.

Elle est donc essentielle pour l'équilibrage des données et la performance.

8. Quels sont les critères de choix d'une bonne clé de sharding ?

Il y a :

- Une haute cardinalité, c'est-à-dire que la clé doit avoir une grande quantité de valeurs différentes.
- Une distribution uniforme, afin que les *shards* puissent être relativement de la même taille.
- Une utilisation fréquente dans les requêtes. Une requête ne pouvant pas profiter de la *clé de sharding* devra parcourir tous les shards, ce qui est coûteux.
- Elle ne doit pas être monotone (valeur augmentant à chaque fois). Le détail est fait en question 14.

9. Qu'est-ce qu'un chunk dans MongoDB ?

Dans MongoDB, un *chunk* représente un intervalle contigu de valeurs de la *clé de sharding*.

10. Comment fonctionne le splitting des chunks ?

Lorsqu'un *chunk* arrive à saturation, MongoDB le divise automatiquement en deux, créant ainsi deux *chunks* plus petits.

11. Que fait le balancer dans un cluster shardé ?

Il assure une distribution équilibrée des chunks. Il déplace les chunks d'un shard à l'autre si besoin pour que tous les shards aient une taille à peu près égale.

12. Quand et comment le balancer déplace-t-il des chunks ?

Le *balancer* tourne en arrière-plan. S'il détecte un déséquilibre dans la taille des *shards*, il prend la décision de déplacer un *chunk* d'un *shard* surchargé à un autre moins chargé.

Les documents sont d'abord copiés vers le *shard* destination, les opérations dans l'*oplog* du *shard* source sont aussi refaites sur le *shard* destination. Le *chunk* est ensuite verrouillé le temps de mettre à jour les données dans le *config server* (le *chunk* vient de changer de *shard*). Enfin, les données sont supprimées du *shard* source.

13. Qu'est-ce qu'un hot shard et comment l'éviter ?

Un *hot shard* désigne un shard recevant trop d'écritures ou de lectures.

Il provient souvent d'une mauvaise décision au niveau du choix de la *clé de sharding*.

14. Quels problèmes une clé de sharding monotone peut-elle engendrer ?

Une *clé de sharding* monotone est une clé qui varie toujours dans la même direction avec chaque nouveau document, comme un identifiant qui s'incrémente automatiquement ou un timestamp.

Avec une clé de ce type, toutes les insertions vont dans le même *shard*, créant ainsi un hot *shard* et un déséquilibre dans la taille des différents *shards*.

15. Comment activer le sharding sur une base de données et sur une collection ?

Au préalable, avoir lancé un config server (`mongod --configsvr`) et un routeur (`mongos`).

Il faut se connecter au routeur et faire :

```
sh.enableSharding("<bdd>")
```

```
sh.shardCollection("<bdd>.<coll>", <clé>)
```

Où :

- `<bdd>` désigne la base de données à *sharder*.
- `<coll>` la collection à *sharder*
- `<clé>` la *clé de sharding*, par exemple `{username:1}` pour créer une clé *ranged* sur le champ `username`. Pour créer un *hashed key*, écrire `"hashed"` à la place de `1`.

16. Comment ajouter un nouveau shard à un cluster MongoDB ?

Toujours sur le routeur,

```
sh.addShard("<replica_set>/<hostname><:port>[,<hostname><:port>]")
```

Où:

- `<replica_set>` : nom du *Replica Set*
- `<hostname><:port>` : Nom d'hôte (ou IP) du serveur et son port.
    - Si le *replica set* contient plusieurs serveurs, la liste doit être séparée par des virgules.

17. Comment vérifier l'état du cluster shardé (commandes usuelles) ?

Les commandes usuelles comprennent :

```
sh.status()

db.printShardingStatus()

db.stats()

rs.status()
```

18. Dans quels cas faut-il envisager d'utiliser un hashed sharding key ?

Cela est à considérer dans le cas où :

- On aurait une clé monotone sinon
- On a beaucoup d'écritures
- On a vraiment besoin d'une distribution uniforme

Par sa nature, elle ne peut pas être utilisée efficacement pour des requêtes par intervalle.

19. Dans quels cas faut-il privilégier un ranged sharding key ?

Lorsque l'on fait souvent des requêtes par intervalle, ou si l'on a besoin de faire des agrégats (date, région, etc)

Risque la création de *hot shards*.

20. Qu'est-ce que le zone sharding et quel est son intérêt ?

Mécanisme qui associe des plages de shard key à des shards précis (zones/tags). Intérêt : localisation (réglementation), isolation de charge (premium, etc.).

21. Comment MongoDB gère-t-il les requêtes multi-shards ?

Le routeur envoie la requête aux *shards* concernés. Il agrège les résultats obtenus.

22. Comment optimiser les performances de requêtes dans un environnement shardé ?

Plusieurs choses sont possibles :

- L'utilisation fréquente de la *clé de sharding* dans les requêtes.
- Indexer la clé.
- Surveiller le *balancer*.

23. Que se passe-t-il lorsqu'un shard devient indisponible ?

S'il est disponible autre part grâce à la réplication, la bascule est faite automatiquement.

Sinon, certaines requêtes vont échouer.

24. Comment migrer une collection existante vers un schéma shardé ?

Il suffit de suivre les étapes décrites en 15. MongoDB se charge automatiquement de distribuer les chunks.

25. Quels outils ou métriques utiliser pour diagnostiquer les problèmes de sharding ?

Les outils disponibles sont :

- `sh.status()` dans le shell Mongo,
- `mongostat`, `mongotop` dans un invite de commandes,
- les logs du *balancer*.

On peut se baser sur les métriques suivantes :

- Distribution des *chunks*,
- Latence,

- Taux d’erreurs.