TP3 – Exploration de CouchDB

Le partitionnement (Sharding) sous MongoDB

Nom : Seyfeddine Jouini

---

1. Introduction

Contrairement à la réplication, qui vise principalement la tolérance aux pannes, le sharding permet de distribuer les données et la charge sur plusieurs nœuds. Ce mécanisme devient indispensable lorsque le volume de données ou le nombre de requêtes devient trop important pour un seul serveur.

L'architecture mise en place repose sur trois composants essentiels : - un serveur de configuration (*config server*), - un routeur *mongos*, - deux shards, chacun étant un *replica set*.

---

2. Architecture du cluster shardé

L'architecture globale du système comprend :

2.1 Config Server (CSRS)

Le *Config Server* stocke les métadonnées du cluster shardé : répartition des *chunks*, clés de sharding, zones, etc.

Il constitue un point critique du système et doit obligatoirement être répliqué.

2.2 Mongos (Routeur)

Le *mongos* agit comme un point d'entrée unique pour les clients. Il reçoit les requêtes, consulte les métadonnées du *config server* et redirige les opérations vers les shards concernés.

2.3 Shards

Les shards contiennent les données réelles. Dans ce TP, chaque shard est configuré comme un *replica set*, ce qui permet d'assurer une tolérance aux pannes locale.

Cette architecture permet à MongoDB de répartir automatiquement les données et la charge entre les différents shards.

---

3. Mise en place de l'environnement

3.1 Préparation

Six terminaux sont ouverts afin de séparer clairement les rôles : - config server, - routeur mongos, - initialisation des replica sets, - shard 1, - shard 2, - client.

Trois répertoires de stockage sont créés : - un pour le config server, - un pour chaque shard.

---

4. Démarrage du cluster MongoDB

4.1 Démarrage du Config Server

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

Même avec un seul nœud, le *config server* est initialisé comme un *replica set*, conformément aux exigences de MongoDB.

4.2 Démarrage du routeur mongos

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

Le *mongos* se connecte au *config server* pour récupérer les informations de configuration du cluster shardé.

4.3 Démarrage des shards

```
mongod --replSet replicashard1 --dbpath serv1/ --shardsvr --port 20004
mongod --replSet replicashard2 --dbpath serv2/ --shardsvr --port 20005
```

Chaque shard dispose : - d'un *replica set* dédié, - d'un répertoire de données propre, - d'un rôle explicite de *shard server*.

4.4 Ajout des shards au cluster

Depuis le routeur *mongos* :

```
sh.addShard("replicashard1/localhost:20004");
sh.addShard("replicashard2/localhost:20005");
```

À ce stade, le cluster shardé est fonctionnel mais aucune base de données n'est encore partitionnée.

---

5. Activation du sharding

5.1 Activation sur la base de données

```
sh.enableSharding("mabasefilms");
```

Par défaut, MongoDB ne shard aucune base sans action explicite de l'administrateur.

5.2 Activation sur la collection

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

La clé de sharding détermine : - la distribution des documents, - le routage des requêtes, - l'équilibrage de la charge.

---

6. Insertion des données et observation

Un programme Python fourni par le cours est exécuté afin d'insérer un grand nombre de documents (jusqu'à un million de films).

Chaque document est inséré individuellement. Au fur et à mesure des insertions : - les *chunks* sont créés, - les *chunks* trop volumineux sont splittés, - le *balancer* migre automatiquement les chunks entre les shards afin d'équilibrer la charge.

---

7. Réponses aux questions

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

Le sharding est un mécanisme de partitionnement horizontal des données qui répartit les documents d'une collection sur plusieurs shards.

Il permet : - de gérer de très grands volumes de données, - de répartir la charge de lecture et d'écriture, - d'assurer la scalabilité horizontale.

2. Différence entre sharding et réplication

- **Réplication** : copie les mêmes données sur plusieurs nœuds pour la tolérance aux pannes.
- **Sharding** : répartit les données entre plusieurs nœuds pour la performance et la scalabilité.

Les deux mécanismes sont complémentaires.

3. Composants d'une architecture shardée

- Config servers (CSRS)
- Mongos

- Shards (souvent sous forme de replica sets)

4. Rôle des config servers

Ils stockent les métadonnées de sharding et sont indispensables au fonctionnement du cluster.

5. Rôle du mongos

Le *mongos* route les requêtes vers les shards concernés et agrège les résultats.

6. Décision du shard de stockage

MongoDB utilise la clé de sharding pour identifier le chunk correspondant et le shard associé.

7. Clé de sharding

Champ utilisé pour répartir les documents entre les shards.

8. Critères d'une bonne clé de sharding

- forte cardinalité
- répartition uniforme
- usage fréquent dans les requêtes
- absence de monotonie

9. Chunk

Un chunk est une unité logique représentant un intervalle de valeurs de la clé de sharding.

10. Splitting des chunks

Lorsqu'un chunk dépasse une taille seuil, il est automatiquement divisé.

11. Rôle du balancer

Il équilibre la répartition des chunks entre les shards.

12. Déplacement des chunks

Effectué automatiquement lorsque la répartition est déséquilibrée.

13. Hot shard

Shard surchargé, évité par un bon choix de clé de sharding.

14. Problèmes d'une clé monotone

Elle concentre les écritures sur un seul shard.

15. Activation du sharding

```
sh.enableSharding("nomBase");
sh.shardCollection("nomBase.collection", { champ: 1 });
```

16. Ajouter un nouveau shard

```
sh.addShard("replicaSet/host:port");
```

17. Vérifier l'état du cluster

```
sh.status();
db.stats();
db.collection.stats();
```

18. Quand utiliser une clé hashée

Pour répartir uniformément des insertions séquentielles.

19. Quand utiliser une clé par plage

Lorsque les requêtes par intervalle sont fréquentes.

20. Zone sharding

Permet d'associer des plages de données à des shards spécifiques.

21. Requêtes multi-shards

Les requêtes sont envoyées aux shards concernés puis agrégées par *mongos*.

22. Optimisation des performances

- bonne clé de sharding
- indexation
- limitation des requêtes multi-shards

23. Shard indisponible

Les données deviennent inaccessibles si le shard n'est pas répliqué.

24. Migration d'une collection existante

MongoDB redistribue progressivement les données après activation du sharding.

25. Outils et métriques de diagnostic

- sh.status()

• logs MongoDB
• métriques système
• outils de monitoring