Base De Données NoSQL

Réalisé par : Meziane Sarah
INFOA3-25/26

Le partitionnement (Sharding) sous MongoDB

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 est 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 :

- **Config Server (CSRS)**
  Il 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.
- **Mongos (router)**
  Il 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.
- **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

Le serveur de configuration est lancé avec la commande suivante :

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

Le routeur est lancé avec :

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

Deux shards sont démarrés, chacun sous forme de replica set :

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, les shards sont ajoutés au cluster :

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

Le sharding est activé explicitement sur la base `mabasefilms` :

sh.enableSharding("mabasefilms")

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

5.2 Activation sur la collection

La collection `films` est shardée à l'aide de la clé `titre` :

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 pour équilibrer la charge.

Réponse aux questions

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

Le **sharding** est un mécanisme de **partitionnement horizontal** des données qui consiste à répartir les documents d’une collection sur plusieurs serveurs appelés *shards*.
Il est utilisé pour :

- gérer de **très grands volumes de données**,
- répartir la **charge de lecture et d’écriture**,
- permettre le **passage à l’échelle horizontale** (scalabilité).

---

2. Différence entre sharding et réplication

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

Les deux mécanismes sont **complémentaires**.

---

3. Composants d’une architecture shardée

- **Config servers (CSRS)** : stockent les métadonnées du cluster.
- **Mongos** : routeur qui reçoit les requêtes clientes.
- **Shards** : serveurs qui stockent les données (souvent sous forme de replica sets).

---

4. Rôle des config servers

Les config servers :

- stockent les **métadonnées de sharding** (chunks, clés, zones),
- indiquent où se trouvent les données,
- sont indispensables au fonctionnement du cluster.

S’ils tombent en panne, le cluster devient inopérant.

---

5. Rôle du mongos

Le **mongos** est un **routeur intelligent** :

- il reçoit les requêtes clientes,
- consulte les config servers,
- redirige les requêtes vers le(s) shard(s) concerné(s),
- agrège les résultats avant de les renvoyer au client.

---

6. Décision du shard de stockage

MongoDB utilise la **clé de sharding** :

- il calcule l'intervalle (ou le hash),
- identifie le chunk correspondant,
- stocke le document sur le shard qui possède ce chunk.

---

7. Clé de sharding

La **clé de sharding** est un champ (ou ensemble de champs) utilisé pour répartir les documents entre les shards.
Elle est essentielle car elle détermine :

- la distribution des données,
- l'équilibrage de la charge,
- l'efficacité des requêtes.

---

8. Critères d'une bonne clé de sharding

Une bonne clé doit :

- avoir une **forte cardinalité**,
- répartir uniformément les écritures,
- être fréquemment utilisée dans les requêtes,
- éviter les valeurs monotones.

---

9. Chunk

Un **chunk** est une unité logique de données représentant un intervalle de valeurs de la clé de sharding, stockée sur un shard.

---

10. Splitting des chunks

Lorsqu'un chunk dépasse une taille seuil :

- MongoDB le **divise en deux chunks plus petits**,
- ces chunks peuvent ensuite être déplacés vers d'autres shards.

---

11. Rôle du balancer

Le **balancer** :

- surveille la répartition des chunks,
- déplace les chunks entre shards,
- assure un **équilibrage de la charge**.

---

12. Déplacement des chunks par le balancer

Le balancer agit :

- automatiquement,
- lorsque la répartition est déséquilibrée,
- principalement en période de faible activité pour limiter l'impact.

---

13. Hot shard

Un **hot shard** est un shard surchargé (écritures ou lectures excessives).
On l'évite en :

- choisissant une clé de sharding adaptée,
- évitant les clés monotones,
- utilisant des clés hashées si nécessaire.

---

14. Problèmes d'une clé monotone

Une clé monotone (ex : timestamp croissant) :

- concentre les écritures sur un seul shard,
- provoque un hot shard,
- réduit les performances globales.

---

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

Commandes usuelles :

sh.status()

db.stats()

db.collection.stats()

---

18. Quand utiliser une clé hashée

Une **hashed sharding key** est utile lorsque :

- les insertions sont séquentielles,
- on veut une distribution uniforme des écritures,
- les requêtes par plage sont peu fréquentes.

---

19. Quand utiliser une clé par plage (ranged)

Une **ranged sharding key** est préférable lorsque :

- les requêtes utilisent des **intervalles**,
- l'ordre des données est important,
- la charge reste bien répartie.

---

20. Zone sharding

Le **zone sharding** permet d'associer des plages de données à des shards spécifiques.
Il est utile pour :

- des contraintes géographiques,
- des exigences réglementaires,
- un contrôle fin de la distribution.

---

21. Requêtes multi-shards

MongoDB :

- envoie la requête à tous les shards concernés,
- agrège les résultats via mongos,
- renvoie un résultat unique au client.

---

22. Optimisation des performances

- Choisir une bonne clé de sharding,
- indexer la clé de sharding,
- éviter les requêtes multi-shards,
- surveiller la répartition des chunks.

---

23. Shard indisponible

Si un shard est indisponible :

- les données qu'il contient deviennent inaccessibles,
- le cluster continue partiellement si les shards sont répliqués,
- les requêtes peuvent échouer selon leur portée.

---

24. Migration d'une collection existante

- Activer le sharding sur la base,
- définir la clé de sharding,
- MongoDB répartit progressivement les données existantes.

---

25. Outils et métriques de diagnostic

- `sh.status()`
- logs MongoDB
- métriques CPU, mémoire, I/O
- outils de monitoring (MongoDB Atlas, Prometheus, etc.)