3ème année cycle d'ingénieur – Spécialité Informatique

TP3 – partitionnement

Réalisé par :
ER-RACHDI Ilias

Encadrés par :
M. YOUCEF

Année Scolaire : 2025/2026

1 Introduction

Ce TP est consacré à l'étude du **partitionnement (sharding)** dans MongoDB. Contrairement à la réplication, qui vise principalement la haute disponibilité, le sharding permet de **répartir les données sur plusieurs serveurs** afin de gérer de très grands volumes et d'améliorer les performances.

L'objectif de ce travail est de mettre en place un cluster MongoDB shardé à l'aide de Docker, puis de comprendre le rôle de chaque composant ainsi que les mécanismes de distribution des données.

2 Architecture d'un cluster shardé

L'architecture mise en place repose sur les éléments suivants :

— **Config Server** : conserve les métadonnées du cluster (répartition des chunks, configuration),

— **Shards** : stockent les données effectives, chaque shard contenant une partie du jeu de données,

— **Routeur mongos** : point d'entrée des applications, il redirige les requêtes vers les shards appropriés.

Cette séparation permet à MongoDB de faire évoluer horizontalement le stockage et la charge.

3 Mise en place du cluster

3.1 Configuration Docker

Le cluster est déployé via `docker-compose` et comprend quatre services : un config server, deux shards et un routeur `mongos`.

```
version: "3.8"
services:
configsvr:
image: mongo:7.0
command: mongod --configsvr --replSet replicaconfig --port 27019

shard1:
image: mongo:7.0
command: mongod --shardsvr --replSet replicashard1 --port 27018

shard2:
image: mongo:7.0
command: mongod --shardsvr --replSet replicashard2 --port 27017

mongos:
image: mongo:7.0
command: mongos --configdb replicaconfig/configsvr:27019 --port 27020
```

3.2 Initialisation des Replica Sets

Chaque composant est initialisé comme un *Replica Set*. Par exemple, pour le config server :

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

La même logique est appliquée aux deux shards.

3.3 Ajout des shards au cluster

Depuis le routeur `mongos` :

```
sh.addShard("replicashard1/shard1:27018")
sh.addShard("replicashard2/shard2:27017")
sh.status()
```

4 Activation du sharding

Le sharding est activé sur la base `mabasefilms`, puis sur la collection `films` en utilisant le champ `titre` comme clé de sharding.

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

Une fois l'insertion des données effectuée, la commande `sh.status()` permet d'observer la répartition progressive des chunks entre les shards.

5 Fonctionnement du sharding

5.1 Clé de sharding

La clé de sharding est le critère utilisé par MongoDB pour déterminer sur quel shard un document doit être stocké. Son choix est déterminant pour assurer une bonne répartition des données.

Une bonne clé doit présenter :

— une forte cardinalité,

— une distribution uniforme,

— une absence de caractère monotone,

— une utilisation fréquente dans les requêtes.

5.2 Chunks et splitting

Les données sont organisées en **chunks**, chacun correspondant à une plage de valeurs de la clé de sharding. Lorsqu'un chunk atteint une taille limite (64 Mo par défaut), MongoDB le divise automatiquement en plusieurs chunks plus petits.

5.3 Balancer

Le *balancer* est responsable de l'équilibrage de la charge. Il déplace des chunks entre shards afin de maintenir une distribution homogène des données.

6 Questions et réponses sur le sharding MongoDB

Cette section répond de manière détaillée à l'ensemble des questions posées dans l'énoncé du TP sur le partitionnement MongoDB, en s'appuyant sur les observations réalisées lors de la mise en place du cluster shardé.

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

Le sharding est un mécanisme de partitionnement horizontal qui consiste à répartir les documents d'une collection sur plusieurs serveurs appelés shards. Il est utilisé lorsque le volume de données ou la charge devient trop important pour être géré par une seule machine. Le sharding permet ainsi d'augmenter la capacité de stockage globale et d'améliorer les performances en parallélisant les accès.

Question 2 : Différence entre sharding et réplication

La réplication vise la haute disponibilité en dupliquant les mêmes données sur plusieurs nœuds. Le sharding, au contraire, divise les données entre plusieurs shards. En résumé, la réplication assure la tolérance aux pannes tandis que le sharding permet le passage à l'échelle.

Question 3 : Composants d'une architecture shardée

Une architecture shardée repose sur trois composants principaux : les shards, qui stockent les données ; les config servers, qui conservent les métadonnées de partitionnement ; et les routeurs mongos, qui servent de point d'entrée aux applications clientes.

Question 4 : Responsabilités des config servers

Les config servers stockent l'ensemble des métadonnées du cluster shardé. Ils maintiennent notamment la correspondance entre les chunks et les shards, ainsi que la configuration globale du cluster. Sans eux, le routeur mongos ne peut pas localiser les données, ce qui rend le cluster inutilisable.

Question 5 : Rôle du routeur mongos

Le mongos est le point d'accès pour les clients. Il analyse chaque requête, consulte les config servers pour déterminer où se trouvent les données, puis redirige la requête vers le ou les shards concernés. En cas de requête multi-shards, il agrège les résultats avant de les renvoyer au client.

Question 6 : Décision du shard de stockage d'un document

MongoDB utilise la valeur de la clé de sharding du document. Cette valeur permet de déterminer le chunk auquel appartient le document, et donc le shard sur lequel il sera stocké.

Question 7 : Clé de sharding et importance

La clé de sharding est le champ utilisé pour partitionner les données. Elle est essentielle car elle conditionne la distribution des documents, l'équilibrage de la charge et l'efficacité des requêtes.

Question 8 : Critères d'une bonne clé de sharding

Une bonne clé de sharding doit avoir une forte cardinalité, une distribution uniforme, ne pas être monotone et être fréquemment utilisée dans les requêtes.

Question 9 : Définition d'un chunk

Un chunk est une plage de valeurs de la clé de sharding. Chaque chunk regroupe une portion des documents et est stocké sur un shard donné.

Question 10 : Fonctionnement du splitting

Lorsque la taille d'un chunk dépasse un seuil prédéfini, MongoDB le divise automatiquement en plusieurs chunks plus petits afin de maintenir des tailles équilibrées.

Question 11 : Rôle du balancer

Le balancer est chargé d'équilibrer la répartition des chunks entre les shards. Il évite qu'un shard ne soit surchargé par rapport aux autres.

Question 12 : Déplacement des chunks par le balancer

Le balancer se déclenche lorsque l'écart de charge entre shards dépasse un certain seuil. Il migre alors des chunks d'un shard surchargé vers un shard moins chargé.

Question 13 : Hot shard

Un hot shard est un shard recevant une charge excessive de requêtes. Il apparaît généralement à cause d'une mauvaise clé de sharding. L'utilisation d'une clé mieux distribuée ou du hashed sharding permet de l'éviter.

Question 14 : Problèmes liés à une clé monotone

Une clé monotone entraîne une concentration des écritures sur un même shard, ce qui crée un déséquilibre et dégrade les performances.

Question 15 : Activation du sharding

Le sharding s'active d'abord sur une base de données, puis sur une collection spécifique à l'aide des commandes extttsh.enableSharding et extttsh.shardCollection.

Question 16 : Ajout d'un shard

Un nouveau shard est ajouté au cluster via la commande extttsh.addShard. Le balancer redistribue ensuite automatiquement les chunks.

Question 17 : Vérification de l'état du cluster

L'état du cluster peut être suivi avec des commandes comme extttsh.status(), extttdb.stats() ou extttsh.getBalancerState().

Question 18 : Utilisation du hashed sharding

Le hashed sharding est recommandé lorsque la clé est monotone, car le hachage permet une distribution uniforme des données.

Question 19 : Utilisation du ranged sharding

Le ranged sharding est préférable lorsque les requêtes portent fréquemment sur des plages de valeurs, par exemple des dates ou des zones géographiques.

Question 20 : Zone sharding

Le zone sharding permet d'associer certaines plages de données à des shards spécifiques, notamment pour des contraintes réglementaires ou de performance.

Question 21 : Gestion des requêtes multi-shards

Si une requête ne contient pas la clé de sharding, le mongos effectue un extitscatter-gather en interrogeant tous les shards, ce qui est plus coûteux.

Question 22 : Optimisation des performances

Les performances peuvent être améliorées en incluant la clé de sharding dans les requêtes, en créant des index adaptés et en limitant les données retournées.

Question 23 : Indisponibilité d'un shard

Si un shard devient indisponible, les données qu'il contient ne sont plus accessibles. Le reste du cluster continue toutefois de fonctionner.

Question 24 : Migration d'une collection existante

La migration vers le sharding consiste à créer un index sur la clé choisie, activer le sharding sur la base, puis shard(er) la collection.

Question 25 : Outils de diagnostic

Le diagnostic du sharding repose sur des commandes comme extttsh.status(), l'analyse des logs, extttdb.currentOp() et des outils graphiques tels que MongoDB Compass.