
TP3 MongoDB
Compte rendu du TP

Réalisé par :

Célian WIRTZ (celian.wirtz@edu.univ-paris13.fr)

3ème année

Année universitaire : 2025/2026

TP3

Partie 1

1. Config Server

1.1 Lancer le processus de configuration

Dans le serveur de configuration, on démarre mongod en mode config server en précisant le répertoire de données cf_serv et le port 27019 :

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

1.2 Initialisation

Dans le terminal client, on se connecte via mongod :

```
mongosh --port 27019
```

Puis on fait rs.initiate comme au TP2.

2. Routeur

Sur le terminal routeur, on utilise mongos en lui indiquant le replica set de config :

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

3. Les shards

Dans le terminal pour le premier shard, on lance mongod en mode shard server en utilisant le répertoire dossier_shard1 et le port 20004 :

```
mongod --replSet repshard1 --dbpath dossier_shard1 --shardsvr --port 20004
```

Puis :

```
mongosh --port 20004
```

```
rs.initiate({  
  _id: "repshard1",  
  members: [{ _id: 0, host: "localhost:20004" }]  
})
```

Bon pour le shard2 on fait pareil...

Dans le terminal du routeur :

```
sh.addShard("repshard1/localhost:20004")  
sh.addShard("repshard2/localhost:20005")
```

5. Activer le sharding pour la base de données

5.1 Autoriser le sharding sur la base de données

Activez le sharding pour la base mabasefilms :

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

5.2 Sharder la collection

On fait :

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

Enfin on exécute le script Python qui insère un grand nombre de documents films dans mabasefilms.films.

Pour avoir un aperçu global du cluster on peut utiliser sh.status().

Partie 2

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

Le sharding est le partitionnement horizontal des données : MongoDB répartit une collection (jeu de données) en “tranches” et stocke ces tranches sur plusieurs shards (nœuds/replica sets) pour augmenter la capacité de stockage et le débit (scalabilité horizontale). On l'utilise quand une seule instance/replica set ne suffit plus en capacité.

2. Différence entre sharding et réplication :

Différence essentielle entre sharding et réplication

La réplication consiste à avoir plusieurs copies des mêmes données sur différents nœuds pour assurer la disponibilité et répartir les lectures. Le sharding, lui, découpe la base en partitions et place chaque portion sur un ensemble de nœuds distinct, typiquement chaque partition est un replica set. En résumé : la réplication apporte redondance et tolérance aux pannes, le sharding permet d'augmenter la capacité en écriture et en stockage en ajoutant des machines.

3. Composants d'une architecture shardée (mongos, config servers, shards) :

Dans un cluster shardé on trouve trois briques : les shards (où résident effectivement les données — généralement des replica sets), les serveurs de configuration qui gardent la « carte » du cluster, et le routeur (mongos) qui sert d'interface entre les applications et le cluster. Chacune de ces briques a un rôle précis dans la circulation et le placement des données.

4. Responsabilités des config servers (CSRS) :

Les config servers conservent les métadonnées du cluster : la correspondance entre intervalles de shard key (chunks) et shards, les zones, l'état du balancer, etc. Ils gèrent aussi les verrous distribués nécessaires lors de migrations et d'opérations d'administration. Ils sont déployés en replica set pour être résilients ; si les config servers tombent, certaines opérations d'administration et de rééquilibrage ne pourront plus avancer.

5. Rôle du mongos router :

Le mongos reçoit les requêtes des clients et se sert des métadonnées pour savoir où envoyer chaque opération. Il coordonne les lectures/écritures et agrège les réponses quand il faut.

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

Le système regarde la valeur de la shard key du document, trouve l'intervalle (chunk) qui contient cette valeur, puis envoie le document vers le shard qui possède ce chunk. Autrement dit, la shard key détermine la partition et donc l'emplacement physique du document.

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

La shard key est le ou les champs indexés utilisés pour fragmenter une collection. Sa sélection est cruciale : elle conditionne la distribution des données, la capacité à cibler précisément les requêtes (et donc éviter le scatter/gather) et l'efficacité du rééquilibrage. Un mauvais choix peut créer des points chauds et des chunks trop gros.

8. Critères de choix d'une bonne clé de sharding :

Idéalement la clé a beaucoup de valeurs distinctes (haute cardinalité), distribue les données de façon homogène entre shards, correspond aux patterns de requête courants pour permettre des requêtes « ciblées », n'est pas strictement monotone (pour ne pas concentrer tous les inserts), et respecte les contraintes d'indexation/taille/unicité nécessaires.

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

Un chunk représente un intervalle des valeurs de la shard key (par exemple de minKey à X). C'est l'unité que MongoDB déplace d'un shard à l'autre. Chaque shard contient plusieurs chunks.

10. Comment fonctionne le splitting des chunks ?

MongoDB surveille la taille et l'activité des chunks, quand un chunk dépasse un seuil il se divise en deux pour faciliter la redistribution. Il existe aussi des commandes pour forcer un split manuellement (comme `sh.splitAt` ou `sh.splitFind`).

11. Que fait le balancer dans un cluster shardé ?

Le balancer est le processus qui examine la répartition des chunks et planifie des migrations (`moveChunk`) pour lisser l'utilisation entre shards. Il opère en coordination avec les config servers et exécute des migrations atomiques pour déplacer des plages de données.

12. Quand et comment le balancer déplace des chunks ?

Dès que la distribution s'écarte de l'équilibre (selon des politiques internes et des seuils), le balancer planifie des `moveChunk` et transfère les documents d'un shard à un autre en respectant les verrous et la

consistance. L'administrateur peut par ailleurs démarrer/arrêter le balancer si besoin.

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

Un hot shard reçoit une part disproportionnée du trafic ou des écritures, créant un goulot d'étranglement. Les causes habituelles : clé monotone ou faible cardinalité. Pour éviter cela, on choisit une shard key bien distribuée, on peut utiliser du sharding hashé, ou revoir l'architecture d'accès aux données.

14. Problèmes d'une clé de sharding monotone :

Une clé monotone concentre les inserts dans le même chunk (donc même shard) : on perd en parallélisme d'écriture, on risque d'avoir des chunks « jumbo » et la performance s'effondre. C'est à éviter sauf cas très particuliers.

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

Connecté à un mongos, on peut activer le sharding au niveau d'une base puis shard une collection. Par exemple : `sh.enableSharding("maDB");` puis `sh.shardCollection("maDB.maColl", { userId: 1 });` On peut préciser des options comme le hachage, l'unicité ou le nombre initial de chunks.

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

On ajoute un shard avec `sh.addShard` en fournissant le nom du replica set et ses adresses seed, par ex. `sh.addShard("replNew/mongo1:27017,mongo2:27017");` Le shard ajouté est typiquement vide au départ ; ensuite le balancer pourra redistribuer les

chunks vers ce nouveau shard. Le shard ajouté doit être vide (ou le prévoir), l'arrivée d'un shard déclenche ensuite le balancer/resharding pour redistribuer les chunks.

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

Depuis un mongos on peut utiliser `sh.status()` ou `db.printShardingStatus()` pour voir la topologie et les chunks. Pour l'état des replica sets on utilise `rs.status()`. Des outils comme `mongostat`, `mongotop`, `db.serverStatus()`, le profiler et les solutions Cloud (Atlas, Cloud Manager) aident au monitoring.

18. Quand envisager un hashed sharding key ?

On opte pour un shard key hashé si la clé candidate est monotone (par ex. un `_id` séquentiel) et crée des hotspots. Le hashing répartit les valeurs pseudo-aléatoirement, mais il casse la possibilité de faire des requêtes par plage efficaces sur cette clé.

19. Quand privilégier un ranged sharding key (plage) ?

Le sharding en plages est préférable si on a besoin de requêtes par intervalle ou d'une localité des données (par ex. requêtes triées par date). Les ranged keys conservent l'ordre naturel et supportent bien les requêtes de type range, à condition de ne pas générer de hotspots.

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

Les zones permettent d'assigner des sous-ensembles de la plage de shard key à des shards particuliers (par exemple pour placer les données européennes sur des shards localisés en Europe). C'est utile pour

la localisation, respecter des SLA, ou optimiser le placement selon la charge.

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

Si la requête contient la shard key (ou son préfixe pour une clé composée), le routeur envoie l'opération uniquement aux shards concernés — c'est une requête ciblée. Si la shard key n'est pas fournie, mongos doit faire un scatter/gather : interroger tous les shards et agréger les résultats, ce qui devient coûteux à grande échelle.

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

Choisir une shard key qui colle aux requêtes fréquentes pour favoriser les ciblées, maintenir des index pertinents sur chaque shard (dont la shard key), réduire les scatter/gather, surveiller les hotspots et utiliser explain()/le profiler pour diagnostiquer les plans et index.

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

Si un shard (ou son replica set) perd sa majorité, il peut devenir indisponible en écriture ; les données qu'il contient ne seront pas accessibles tant que le replica set n'est pas restauré. Les autres shards peuvent continuer à servir les données qui leur appartiennent, mais les migrations et certaines opérations globales seront impactées. Il faut suivre les procédures de reprise adéquates (reconfig, restauration, etc.). Les config servers doivent rester joignables pour le routage complet.

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

On peut shard une collection vide après l'avoir préparée (avec presplitting si nécessaire), utiliser `reshardCollection` (à partir de la v5.0) pour changer la clé ou transformer une collection existante sans downtime, ou faire un dump/load manuel (plus risqué). Avant tout resharding, tester en staging et vérifier les prérequis.

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

Parmi les outils : `sh.status()`, `db.printShardingStatus()`, `rs.status()`, `mongostat`, `mongotop`, `db.serverStatus()`, logs, profiler et les tableaux de bord Atlas/Cloud Manager.

Les métriques utiles : latences d'opérations, OPS/sec, I/O réseau, répartition des chunks, CPU, espace disque, taux de lock, nombre de migrations échouées, présence de chunks jumbo, et fréquence des scatter/gather.

Ces éléments permettent d'identifier hotspots, migrations bloquées et requêtes coûteuses.