TP4 : Le partitionnement (Sharding) sous MongoDB

1. Objectif du TP

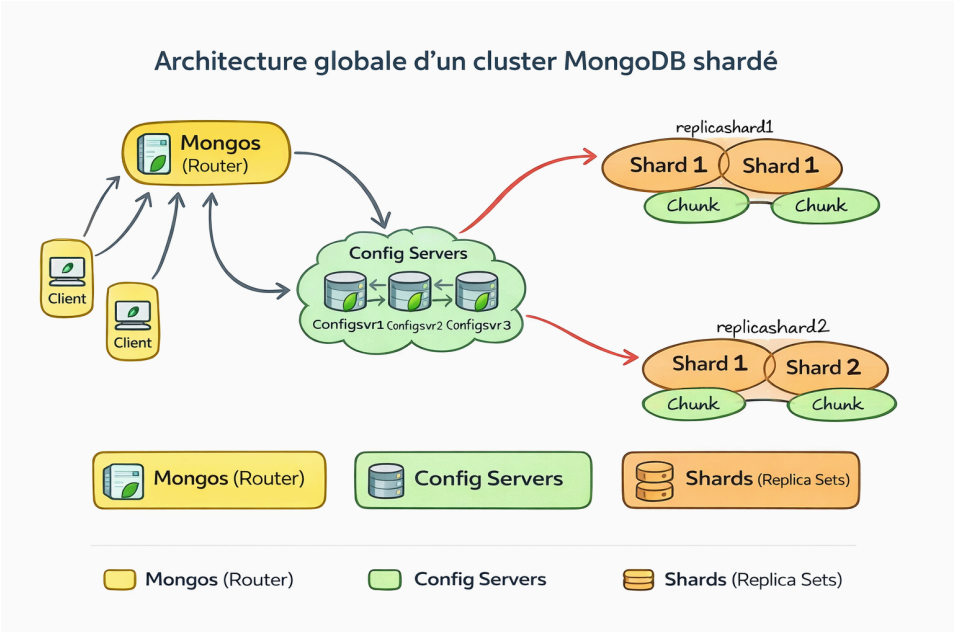
L'objectif de ce TP est de mettre en œuvre une **architecture MongoDB shardée**, afin de comprendre le fonctionnement du **partitionnement des données**, la répartition de la charge, ainsi que le rôle des différents composants d'un cluster MongoDB shardé.

Le TP met en évidence :

- La scalabilité horizontale de MongoDB
- La gestion des volumes importants de données
- L'équilibrage automatique des données entre plusieurs serveurs

---

2. Architecture globale du cluster shardé



Un cluster MongoDB shardé repose sur **trois composants essentiels** :

2.1 Composants de l'architecture

1. **Config Servers (CSRS – Config Server Replica Set)**
    - Stockent les métadonnées du sharding
    - Indispensables au fonctionnement du cluster
    - Répliqués pour assurer la tolérance aux pannes
2. **Shards**
    - Contiennent les données réelles
    - Chaque shard est un **replica set**
    - Les données sont réparties en **chunks**
3. **Mongos (Router)**
    - Point d'entrée des clients
    - Route les requêtes vers les shards appropriés
    - Interroge les config servers pour localiser les données

---

3. Mise en place du cluster MongoDB shardé

3.1 Ouverture des terminaux

Ouvrir **six terminaux** et leur attribuer des noms explicites, par exemple :

- `ConfigSrv`
- `Router`
- `Shard1`
- `Shard2`
- `Client`
- `InitReplica`

3.2 Création des répertoires de stockage

```
mkdir configsvrdb
mkdir serv1
mkdir serv2
```

3.3 Démarrage du Config Server

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

Initialisation du replica set :

```
rs.initiate()
```

3.4 Démarrage du routeur mongos

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

3.5 Démarrage des shards

Shard 1 :

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

Shard 2 :

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

Initialisation des replica sets :

```
rs.initiate()
```

3.6 Ajout des shards au cluster

Connexion au routeur mongos :

```
mongo --port 27017
```

Ajout des shards :

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

---

4. Activation du sharding

4.1 Activation sur la base de données

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

4.2 Sharding de la collection

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

- `titre` est la **clé de sharding**
- Sharding de type **ranged**

---

5. Questions théoriques – Réponses

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

Le sharding est une technique de **partitionnement horizontal** des données permettant de répartir les documents sur plusieurs serveurs afin d’améliorer la **scalabilité** et les **performances**.

---

2. Différence entre sharding et réplication

- **Réplication** : tolérance aux pannes, haute disponibilité

- **Sharding** : montée en charge et distribution des données

---

3. Composants d’une architecture shardée

- **Mongos**
- **Config Servers**
- **Shards**

---

4. Rôle des config servers

Ils stockent :

- Les métadonnées de sharding
- La localisation des chunks
- La configuration du cluster

---

5. Rôle du mongos

Il agit comme **routeur intelligent**, redirigeant les requêtes vers les shards concernés.

---

6. Comment MongoDB choisit le shard d’un document ?

À partir de la **clé de sharding**, MongoDB détermine le chunk cible et donc le shard correspondant.

---

7. Qu’est-ce qu’une clé de sharding ?

Un champ utilisé pour distribuer les documents dans le cluster.

---

8. Critères d’une bonne clé de sharding

- Forte cardinalité
- Distribution uniforme
- Présente dans les requêtes fréquentes
- Non monotone

---

9. Qu’est-ce qu’un chunk ?

Un chunk est une **plage de valeurs de la clé de sharding** stockée sur un shard.

---

10. Fonctionnement du splitting

Lorsqu’un chunk dépasse une taille limite, il est automatiquement **divisé en deux**.

---

11. Rôle du balancer

Il assure une **répartition équilibrée des chunks** entre les shards.

---

12. Déplacement des chunks

Le balancer déplace les chunks automatiquement lorsque le cluster est déséquilibré.

---

13. Hot shard

Shard recevant trop de requêtes.
Solution : bonne clé de sharding ou clé hashée.

---

14. Problèmes d’une clé monotone

- Concentration des écritures sur un seul shard
- Déséquilibre de charge

---

15. Activation du sharding

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

---

16. Ajout d’un shard

```
sh.addShard("replica/host:port")
```

---

17. Vérification de l’état du cluster

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

---

18. Quand utiliser une clé hashée ?

- Écritures intensives
- Répartition uniforme requise

---

19. Quand utiliser une clé ranged ?

- Requêtes par intervalle
- Analyses chronologiques

---

20. Zone sharding

Permet d’associer des plages de données à des shards spécifiques (ex : par région).

---

21. Requêtes multi-shards

MongoDB interroge plusieurs shards et agrège les résultats.

---

22. Optimisation des performances

- Bon choix de clé
- Indexation adaptée
- Limiter les requêtes multi-shards

---

23. Indisponibilité d'un shard

Si répliqué : bascule automatique
Sinon : données temporairement indisponibles

---

24. Migration vers une collection shardée

- Activer le sharding
- Choisir une clé
- MongoDB redistribue les données automatiquement

---

25. Outils de diagnostic

- `sh.status()`
- Logs MongoDB
- MongoDB Compass
- Métriques serveur (CPU, RAM, I/O)

---

6. Conclusion

Ce TP met en évidence l’importance du **sharding** pour la gestion de grandes volumétries de données. MongoDB offre une solution robuste et automatisée pour la montée en charge horizontale, à condition de choisir judicieusement la clé de partitionnement.