Comprendre le Sharding dans MongoDB

Djebabla Ammar

Université de Technologie - Département Informatique

17 juin 2025

Résumé

Ce document présente une explication approfondie du concept de **sharding** dans MongoDB, incluant son architecture, ses avantages et des cas d'utilisation pratiques. Un TP complet permet de mettre en œuvre ces concepts avec des exemples concrets de répartition de données.

Table des matières

1 Introduction au Sharding

Le **sharding** est une technique fondamentale dans les bases de données distribuées qui permet de partitionner horizontalement les données sur plusieurs serveurs (appelés **shards**). Cette approche est particulièrement utile pour gérer de très grandes collections de données qui dépassent les capacités d'un seul serveur.

Architecture typique d'un cluster MongoDB shardé

2 Architecture détaillée

2.1 Composants principaux

— **mongos** : Le routeur qui dirige les opérations de lecture/écriture vers les shards appropriés
— **Config Servers** : Stockent les métadonnées du cluster (namespace, plages de clés, etc.)
— **Shards** : Serveurs contenant les données partitionnées

2.2 Types de Sharding

1. **Sharding par plage (Range-based)** : Partitionnement basé sur des plages de valeurs
2. **Sharding par hachage (Hash-based)** : Utilisation d'une fonction de hachage pour distribuer les données
3. **Sharding par zone (Zone-based)** : Attribution de plages spécifiques à des shards particuliers

3 Exemple détaillé : Sharding de 1000 clients

3.1 Configuration initiale

```
// Activer le sharding pour la base de données
use admin
sh.enableSharding("testdb")

// Créer la collection shardée
use testdb
db.createCollection("clients")

// Choisir la clé de sharding
sh.shardCollection("testdb.clients", { client_id: 1 })
```

Listing 1 – Initialisation du sharding

Information

Remarque : Le choix de la clé de sharding est crucial. Une bonne clé doit avoir :

— Une cardinalité élevée

— Une distribution uniforme

— Des requêtes qui l'utilisent fréquemment

3.2 Découpage des données

```
// Découper la plage de données en 3 chunks
sh.splitAt("testdb.clients", { client_id: 333 })
sh.splitAt("testdb.clients", { client_id: 666 })

// Répartir les chunks sur les shards
sh.moveChunk("testdb.clients",
  { client_id: 100 }, "shard1")
sh.moveChunk("testdb.clients",
  { client_id: 500 }, "shard2")
sh.moveChunk("testdb.clients",
  { client_id: 800 }, "shard3")
```

Listing 2 – Découpage en chunks

3.3 Insertion des données

```
// Insertion de 1000 documents
for (let i = 1; i <= 1000; i++) {
  db.clients.insertOne({
    client_id: i,
    nom: "Client_" + i,
    email: "client" + i + "@example.com",
    date_creation: new Date(),
    solde: Math.random() * 1000
  })
}
```

Listing 3 – Insertion des documents

3.4 Vérification

```
use config

// Afficher la distribution des chunks
db.chunks.aggregate([
```

```
  { $match: { ns: "testdb.clients" }},
  { $group: {
      _id: "$shard",
      count: { $sum: 1 },
      min: { $min: "$min.client_id" },
      max: { $max: "$max.client_id" }
    }
  }
])
```

Listing 4 – Vérification de la distribution

TABLE 1 – Répartition des données

Shard	Nombre de chunks	Plage min	Plage max
shard1	1	1	332
shard2	1	333	665
shard3	1	666	1000

4 TP : Répartition de données d'étudiants

4.1 Objectifs pédagogiques

— Comprendre la répartition automatique des données
— Maîtriser les commandes de gestion du sharding
— Analyser la distribution des données

4.2 Énoncé de l'exercice

4.2.1 Partie 1 : Configuration initiale

1. Créer une base de données nommée `etudiantsDB` et une collection `notes`.
2. Activer le sharding sur la base de données.
3. Activer le sharding sur la collection `notes` en utilisant `etudiant`$_i d comme cle de sharding avec un hachage. V

4.2.2 Partie 2 : Insertion des données

4. Insérer 9000 documents dans la collection `notes`. Chaque document doit contenir :
   — un identifiant d'étudiant `etudiant`$_i d$, $un nom sous la forme$`Etudiant_X`,
   — deux notes aléatoires : `note_math` et `note_physique`,
   — une promotion calculée à partir de l'identifiant.

4.2.3 Partie 3 : Analyse de la distribution

1. Analyser la répartition des chunks entre les shards à l'aide d'une requête d'agrégation sur la base `config`.
2. Vérifier si le balancer est actif.

Conclusion

Ce TP permet de manipuler les concepts clés du sharding :

— Configuration du sharding
— Insertion massive de données
— Analyse de la répartition et vérification de l'équilibrage

Pour approfondir le sujet, consulter la documentation officielle MongoDB : `https://www.mongodb.com/docs/manual/sharding/`

A Annexe : Commandes utiles à connaître

— `sh.status()` : afficher l'état du sharding
— `sh.enableSharding("maDB")` : activer le sharding pour une base
— `sh.shardCollection("maDB.maCollection", { champ : "hashed" })` : activer le sharding sur une collection
— `sh.moveChunk()` : déplacer manuellement un chunk
— `sh.startBalancer()` / `sh.stopBalancer()` : activer/désactiver le balancer
— `db.collection.getShardDistribution()` : voir la répartition des documents