UNIVERSITÉ
CÔTE D'AZUR

Rapport du projet Bases de Données

Gestion d'une bibliothèque MCD, conversionMCD et Génération des Objets MongoDB

Rédigé par
Hadil AYARI
Nadjib KHAMMAR
Miléna KOSTOV
David PRIGODIN

15 juin 2023

Table des matières

1 Modèles de données supportés

1.1 Modèle orienté document

MongoDB est une base de données orientée document, ce qui signifie qu'elle stocke les données sous forme de documents JSON (JavaScript Object Notation). Chaque document est une unité de données autonome qui peut contenir des champs et des valeurs variés, similaires à des enregistrements ou des lignes dans d'autres systèmes de gestion de bases de données relationnelles.

Dans notre projet, chaque entité a un fichier JSON où sont stockées les informations correspondantes. En théorie, les fichiers JSON peuvent contenir une infinité d'informations structurées.

1.2 Schéma flexible

MongoDB offre une flexibilité dans la définition du schéma des documents. Contrairement aux bases de données relationnelles qui ont des schémas rigides, MongoDB permet d'ajouter, de modifier ou de supprimer des champs au sein d'un document sans nécessiter de modification du schéma global. Cela permet une évolutivité et une adaptation faciles aux changements des exigences de données.

Dans notre projet, contrairement à d'autres projets de réalisation d'une base de données, il n'a pas été nécessaire de créer les champs de chaque entité dans le constructeur. C'est pourquoi il y a plusieurs fonctions qui sont identiques d'une classe à une autre.

1.3 Indexation

MongoDB prend en charge l'indexation pour une recherche rapide et efficace des données. On peut créer ainsi des index sur les champs fréquemment consultés pour améliorer les performances des requêtes.

1.4 Réplication

MongoDB offre des fonctionnalités de réplication qui permettent de maintenir des copies synchronisées des données sur plusieurs serveurs. Cela garantit la disponibilité et la fiabilité des données en cas de panne ou de défaillance d'un serveur.

1.5 Sharding

MongoDB prend en charge le sharding, qui est une technique de partitionnement des données sur plusieurs serveurs. Cela permet de distribuer la charge de travail et de stocker de grandes quantités de données de manière évolutive.

2 Réévaluer la procédure d'installation du moteur et des utilitaires

On utilise PowerShell pour lancer le programme. On utilise un seul fichier JAR qui est le même que celui utilisé durant les TPs. Il n'y a aucune différence d'installation par rapport au TP. On prend le chemin jusqu'au dossier MongoDB, puis on initialise MYPATH avec la commande SET en utilisant le chemin absolu jusqu'au dossier MongoDB. Une fois que cela est fait, on exécute les lignes de commandes suivantes :

```
javac -g -cp %MYPATH%\mongojar\mongo-java-driver-3.12.10.jar;%
    MYPATH% %MYPATH%\MongoDB\nom_classe.java

java -Xmx256m -Xms256m -cp %MYPATH%\mongojar\mongo-java-driver
    -3.12.10.jar;%MYPATH% MongoDB.nom_classe
```

3 Architecture du moteur NoSql (avec des schémas expliqués)

MongoDB adopte une architecture de base de données orientée documents. Les données sont stockées sous forme de documents BSON (Binary JSON), qui sont des représentations binaires de documents JSON.

3.1 Cluster de serveurs

MongoDB peut fonctionner en mode cluster, avec plusieurs serveurs interconnectés pour offrir une haute disponibilité et une tolérance aux pannes. Les serveurs sont répartis en différents rôles tels que les nœuds primaires, les nœuds secondaires et les arbitres.

L'architecture du cluster shard comprend trois composants :

— **Serveur shard :** Le shard est un serveur de base de données qui se compose de différents ensembles de réplicas, chacun contenant une partie de vos données. Chaque ensemble de réplicas ou shard est responsable d'une valeur indexée spécifique dans votre base de données.

— **Routeur de requêtes (mongos) :** Mongos est un processus s'exécutant sur l'application cliente qui communique avec MongoDB. Étant donné que les données sont réparties entre différents shards, la requête doit être acheminée vers le shard qui détient les données demandées. Il est courant d'avoir des routeurs de requêtes distincts pour chaque serveur d'application.

— **Serveur de configuration :** Le serveur de configuration stocke les métadonnées des shards. Mongos communique avec le serveur de configuration pour déterminer quels ensembles de réplicas interroger pour les données. En général, les serveurs de configuration doivent comprendre au moins trois serveurs. Ils n'ont pas de capacité de basculement ; si l'un des serveurs échoue, l'ensemble du processus est hors ligne.

Cf. figure 1.

3.2 Architecture distribuée

MongoDB permet de distribuer les données sur plusieurs serveurs, offrant ainsi une évolutivité horizontale. Les données peuvent être réparties sur différents nœuds en fonction de règles de partitionnement définies.

Cf. figure 2.

3.3 Stockage des données

MongoDB utilise un système de stockage flexible appelé WiredTiger par défaut. WiredTiger prend en charge la compression, la mise en cache en mémoire, l'indexation et d'autres fonctionnalités avancées pour améliorer les performances et l'efficacité du stockage.

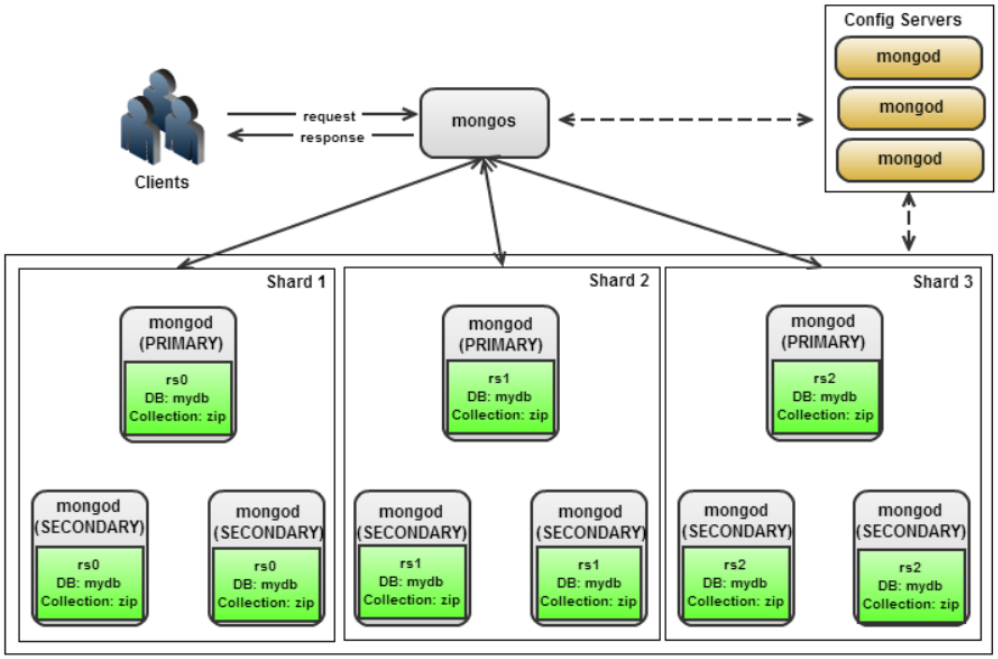

FIGURE 1 – Architecture du cluster shard

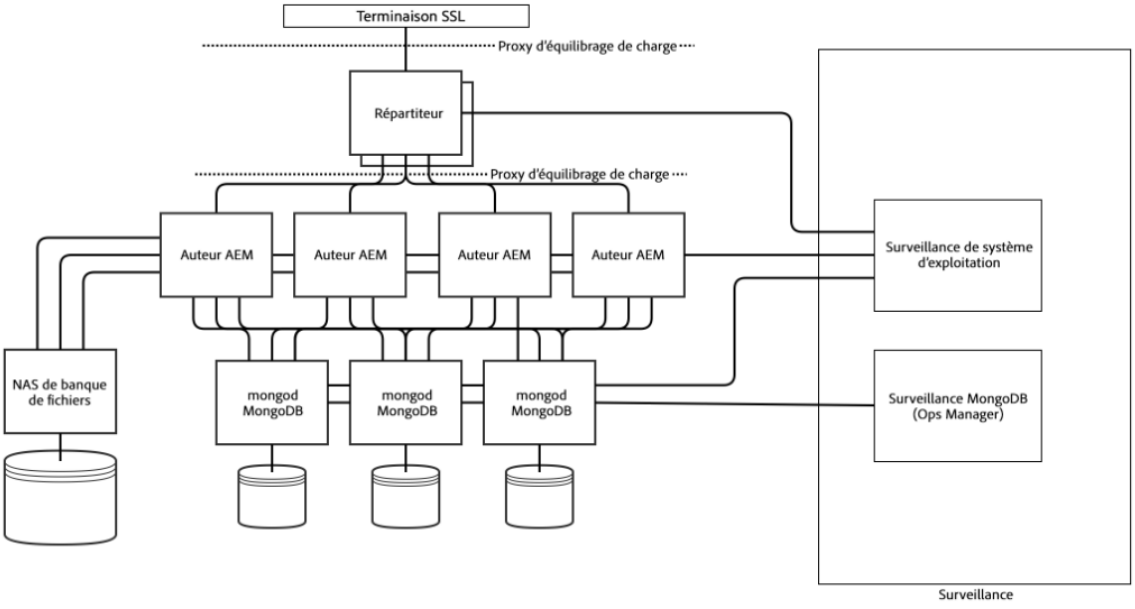

FIGURE 2 – Un déploiement minimal nécessite trois mongod instances configurées en tant qu'ensemble de réplication. Une instance est élue Principale, les autres instances étant secondaires, l'élection étant gérée par mongod. Un disque local est associé à chaque instance. La grappe peut donc prendre en charge la charge.

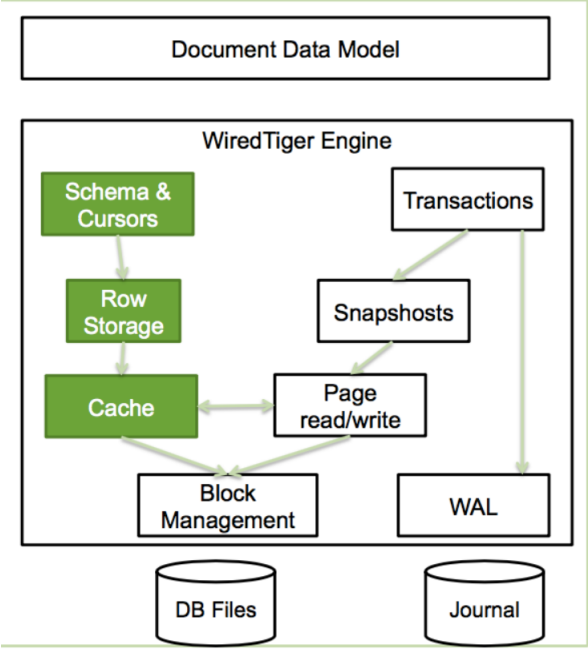

FIGURE 3 – Structure de WiredTiger

WiredTiger est un moteur de stockage utilisé par MongoDB qui offre des fonctionnalités avancées telles que la compression des données, la gestion des transactions, le verrouillage des données, la gestion de la mémoire, la gestion des journaux et le partitionnement des données. Ces fonctionnalités contribuent à améliorer les performances, l'efficacité et la fiabilité des opérations de stockage et de récupération des données.

3.4 Indexation

MongoDB prend en charge différents types d'index, tels que les index simples, les index composés, les index géospatiaux et les index textuels. Les index améliorent les performances des requêtes en accélérant la recherche et en facilitant l'optimisation des requêtes.

Les index MongoDB utilisent une structure de données appelée B-arbre (arbre B en français). Une B-arbre est une structure de données hiérarchique qui permet de stocker et d'organiser efficacement les données pour faciliter les opérations de recherche et de récupération rapides. Dans le contexte de MongoDB, un index B-arbre est utilisé pour accélérer la recherche de documents dans une collection. L'index est construit sur un champ ou un ensemble de champs spécifiés, ce qui permet de localiser rapidement les documents correspondants lors d'une requête.

Cf. figures 4 et 5.

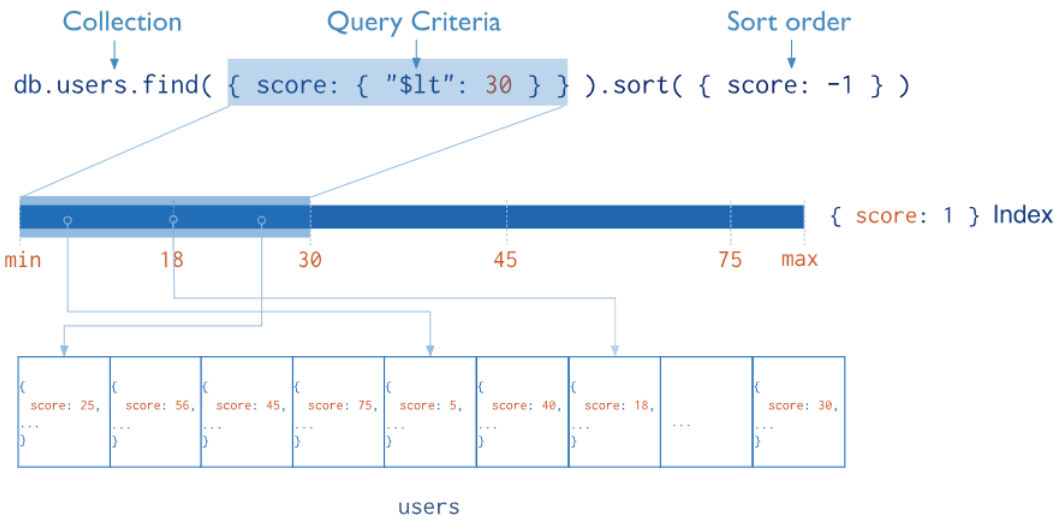

FIGURE 4 – Structure de données appelée B-arbre

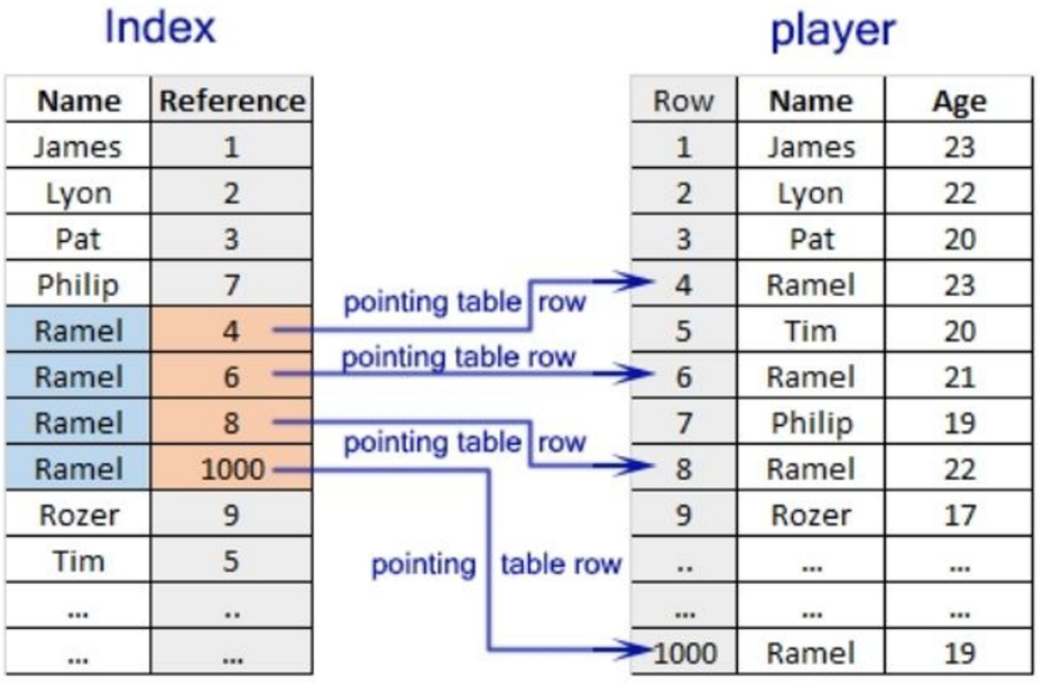

FIGURE 5 – L'utilisation des index permet de retrouver plus vite les informations concernant Ramel

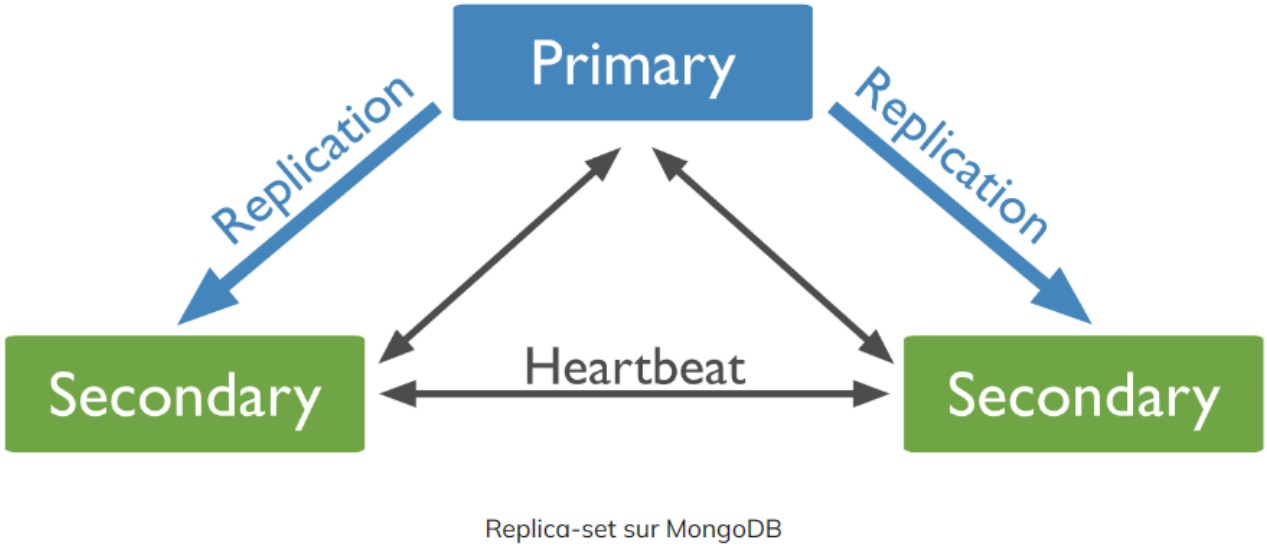

FIGURE 6 – Si l'instance leader, appelée Primary est en panne, les autres instances vont élire un nouveau leader et la base de données sera indisponible pendant ce processus (de quelques secondes). La garantie que toute requête aura une réponse n'est pas assurée.

3.5 Réplication

MongoDB offre la possibilité de répliquer les données sur plusieurs serveurs pour assurer la disponibilité et la tolérance aux pannes. La réplication se fait en temps réel, où les modifications effectuées sur un nœud sont automatiquement propagées aux autres nœuds du cluster.

Cf. figure 6.

3.6 Sharding

MongoDB prend en charge le sharding, qui permet de distribuer les données sur plusieurs serveurs en fonction de règles de partitionnement. Cela permet de gérer des volumes de données massifs et d'améliorer les performances en parallélisant les opérations de lecture/écriture sur différents nœuds.

Cf. figure 7.

3.7 Gestion de la mémoire

MongoDB dispose d'un gestionnaire de cache intégré qui optimise l'utilisation de la mémoire pour améliorer les performances. Il utilise des stratégies de mise en cache basées sur les modèles d'accès aux données pour stocker les données fréquemment utilisées en mémoire.

3.8 Architecture flexible

MongoDB offre une grande flexibilité en permettant l'ajout de nouveaux champs aux documents sans avoir à définir de schéma rigide à l'avance. Cela facilite l'évolution de la structure des

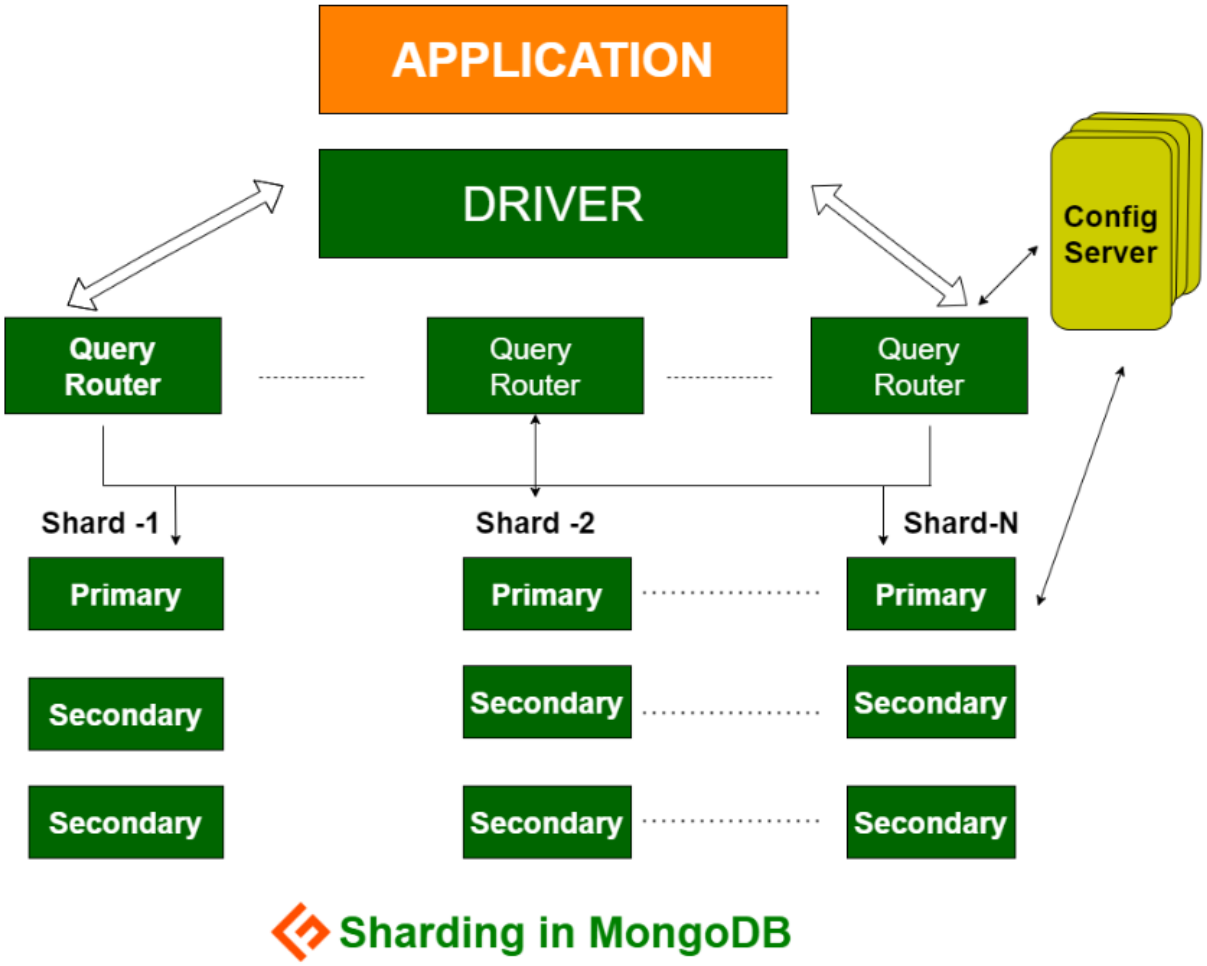

FIGURE 7 – Chaque jeu de réplicas comprend au moins 3 instances mongo ou plus. Un cluster partitionné peut être constitué de plusieurs instances de partitions mongo, et chaque instance de partition fonctionne dans un jeu de réplicas de partition. L'application interagit avec Mongos, qui à son tour communique avec tessons. Par conséquent, dans le partage, les applications n'interagissent jamais directement avec les nœuds de partition. Le routeur de requête distribue les sous-ensembles de données entre les nœuds de partition en fonction de la clé de partition.

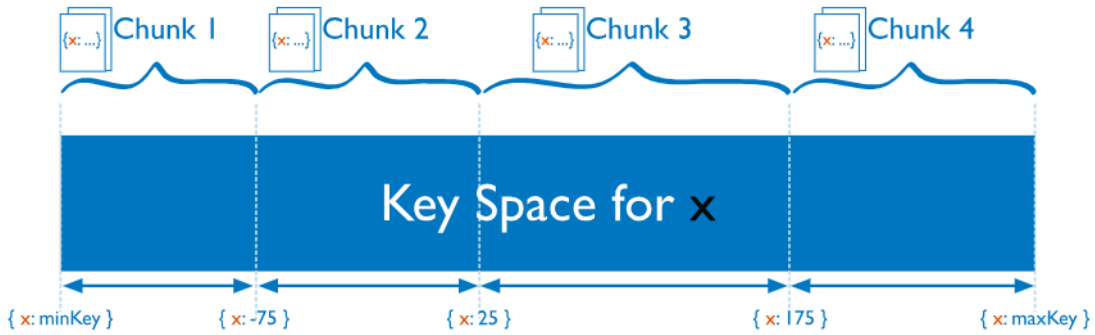

FIGURE 8 – Partitionnement basé sur une plage (Range-based Sharding)

données au fil du temps.

4 Méthodes de partitionnement

Le partitionnement est une technique utilisée dans MongoDB pour gérer des ensembles de données volumineux et les répartir sur plusieurs serveurs. Cela permet d'améliorer les performances et la capacité de stockage, tout en assurant une haute disponibilité des données.

Dans MongoDB, le partitionnement se fait en divisant les données en morceaux appelés "shards" et en les distribuant sur différents serveurs appelés "nœuds de shard". Chaque shard contient un sous-ensemble des données, ce qui permet une répartition équilibrée de la charge de travail.

Il existe 2 méthodes de partitionnement utilisées par MongoDB :

4.1 Partitionnement basé sur une plage (Range-based Sharding)

Le partitionnement basé sur une plage (Range-based Sharding) divise les données en plages contiguës déterminées par les valeurs de la clé de partitionnement. Les documents ayant des valeurs de clé proches sont regroupés dans le même fragment ou shard, ce qui facilite les requêtes efficaces sur des plages de données spécifiques. Cependant, une mauvaise sélection de la clé de partitionnement peut affecter les performances en lecture et en écriture. Le partitionnement basé sur une plage est la méthode par défaut si aucune autre option comme le partitionnement par hachage (Hashed Sharding) ou par zones (Zone Sharding) n'est configurée. Dans ce type de partitionnement, les données sont réparties en fonction des valeurs d'une clé, souvent une clé d'index. Chaque partition (shard) est responsable d'un intervalle de valeurs, ce qui permet une répartition équilibrée des données entre les shards.

Cf. figure 8.

4.2 Partitionnement basé sur un hachage (Hash-based Sharding)

Le partitionnement par hachage (Hashed Sharding) permet une répartition équilibrée des données à travers le cluster partitionné, mais cela réduit les opérations ciblées au profit des opérations de diffusion. Les documents avec des valeurs de clé de partitionnement similaires sont généralement répartis de manière aléatoire sur les shards, ce qui peut nécessiter des opérations

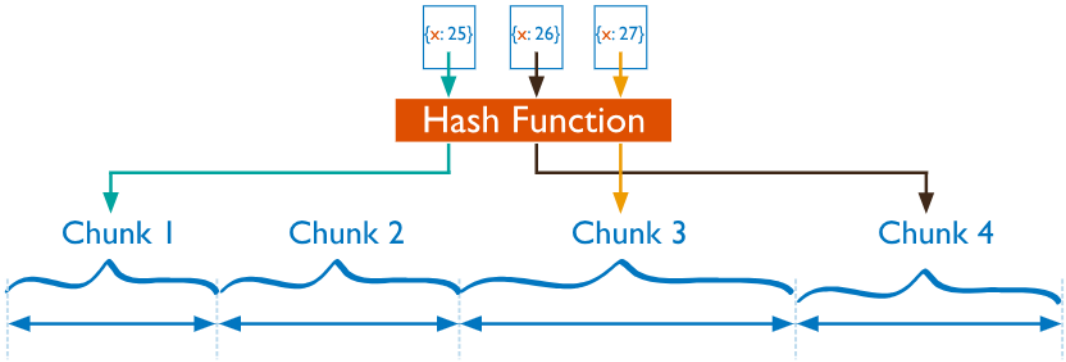

FIGURE 9 – Partitionnement basé sur un hachage (Hash-based Sharding)

de diffusion pour répondre à certaines requêtes. Cette méthode est efficace pour les requêtes avec des correspondances d'égalité, mais peut rendre les requêtes basées sur des plages de valeurs plus complexes.

Cf. figure 9.

5 Méthode de réplication

MongoDB utilise principalement une méthode de réplication appelée "ensembles de réplicas" (replica sets). Il y a différentes fonctionnalités et aspects liés à la réplication au sein des ensembles de réplicas.

5.1 Ensembles de réplicas

MongoDB utilise une architecture de réplication basée sur des ensembles de réplicas, où un ensemble de serveurs MongoDB contient des nœuds primaires et secondaires répliquant les mêmes données.

5.2 Élection automatique

En cas de défaillance du nœud primaire, une élection automatique se produit pour choisir un nouveau nœud primaire parmi les nœuds secondaires disponibles.

Cf. figure 10.

5.3 Réplication asynchrone

Les opérations sont répliquées de manière asynchrone des nœuds primaires vers les nœuds secondaires, ce qui peut entraîner un certain délai de réplication.

Cf. figure 11.

Dans ce scénario, beaucoup plus rapide pour le client, deux phénomènes apparaissent :

— le client reçoit un acquittement alors que la réplication n'est pas complète ; il n'y a donc pas à ce stade de garantie complète de sécurité ;

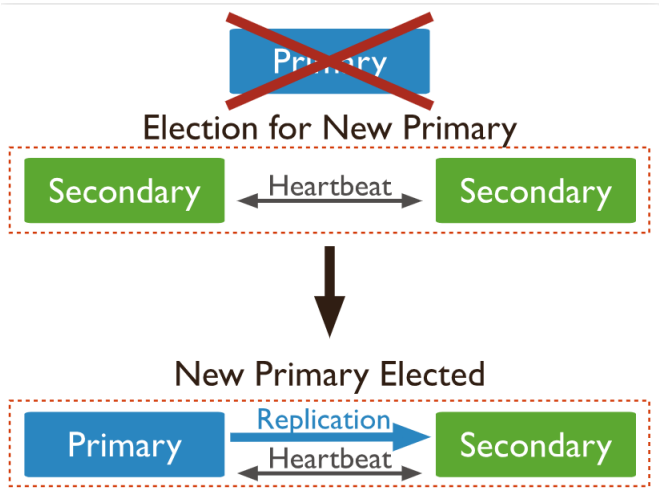

FIGURE 10 – Exemple d'une élection

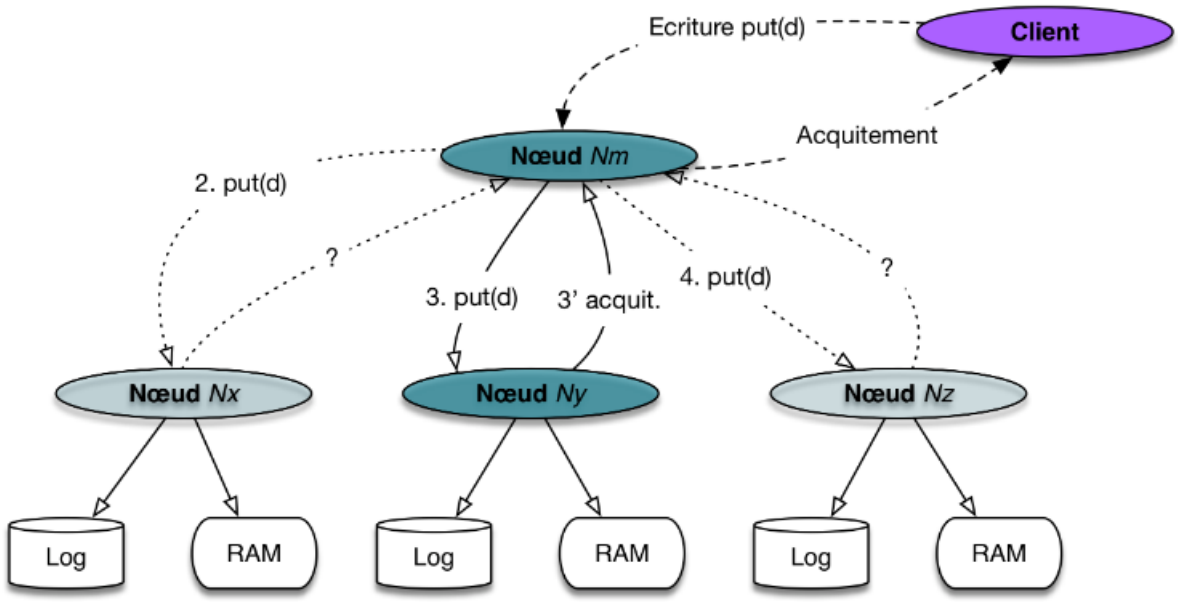

FIGURE 11 – Réplication avec écritures asynchrones

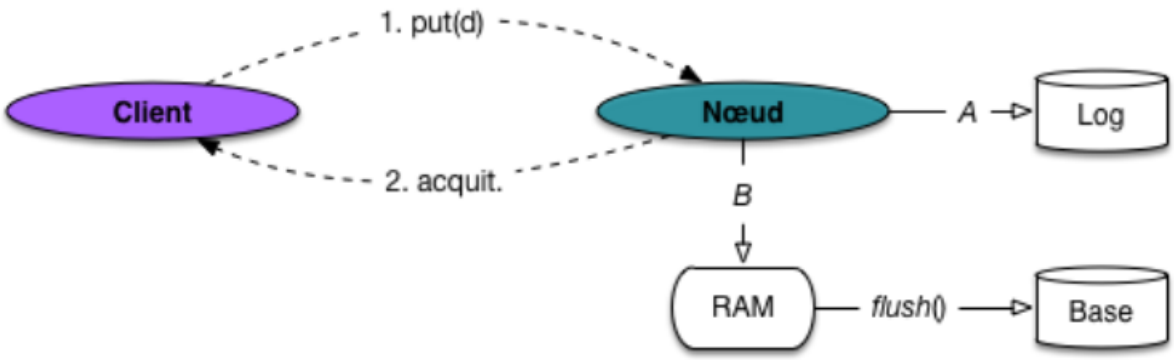

FIGURE 12 – Ecriture avec journalisation

— le client poursuit son exécution alors que toutes les copies de ne sont pas encore mises à jour ; il se peut alors qu'une lecture renvoie une des versions antérieures de d.

Il y a donc un risque pour la cohérence des données.

5.4 Lecture depuis les nœuds secondaires

Les applications peuvent lire à partir des nœuds secondaires, ce qui répartit la charge de lecture et améliore les performances du système.

5.5 Journaling

MongoDB utilise le journaling pour garantir la durabilité des opérations, en enregistrant les opérations dans un journal avant de les répliquer.
Cf. figure 12.

5.6 Surveillance et administration

MongoDB fournit des outils et des fonctionnalités pour surveiller et administrer les ensembles de réplicas. On peut citer comme surveillance :

— Statistiques de performance
— Utilisation des ressources (utilisation du processeur, mémoire disponible et utilisation du réseau)
— Statistiques d'affirmation
— Statistiques de réplication
— Saturation des ressources
— Opérations de débit

6 Montée en charge

La montée en charge, également appelée scalabilité, fait référence à la capacité d'un système informatique à gérer efficacement une augmentation de la charge de travail ou du nombre d'uti-

FIGURE 13 – Montée en charge horizontale

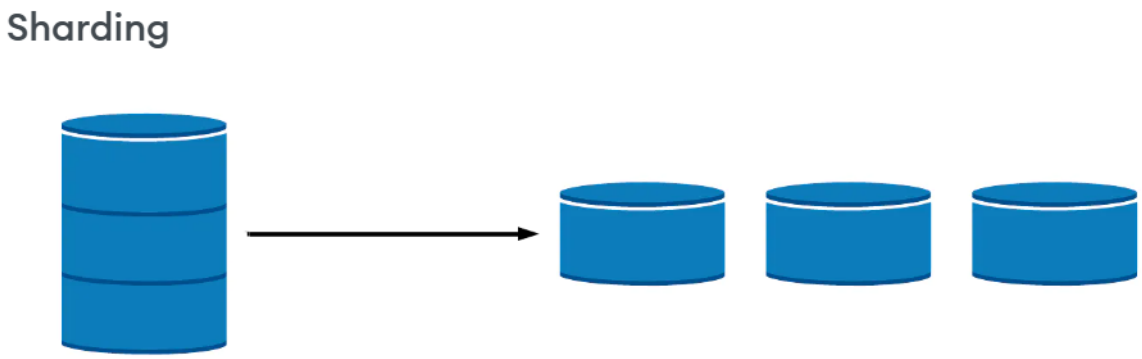

FIGURE 14 – Configuration de la clé de partitionnement dans MongoDB Atlas pour un partitionnement automatique et non chevauchant des données entre les shards. Options de partitionnement : plage, hachage ou zones personnalisées

lisateurs. Cela implique de pouvoir faire face à une augmentation de la demande de manière à maintenir les performances et la disponibilité du système.

Dans le contexte de MongoDB, la montée en charge se réfère à la capacité du système à gérer une augmentation du volume de données, du trafic des requêtes et du nombre de connexions simultanées. L'objectif est de garantir que MongoDB peut s'adapter et maintenir des performances optimales, même lorsque le système est soumis à des charges de travail élevées.

La principale méthode de montée en charge utilisée est la montée en charge horizontale.

La montée en charge horizontale consiste à ajouter des nœuds supplémentaires pour partager la charge. Avec les bases de données non relationnelles, cela est plus simple car les collections sont autonomes et ne sont pas liées de manière relationnelle. Cela leur permet d'être distribuées plus simplement sur plusieurs nœuds, car les requêtes n'ont pas besoin de les "joindre" ensemble sur différents nœuds.

La montée en charge horizontale de MongoDB est réalisée grâce au partitionnement (sharding) (préféré) et aux ensembles de réplicas (replica sets).

Cf. figure 13.

Cf. figures 14 et 15.

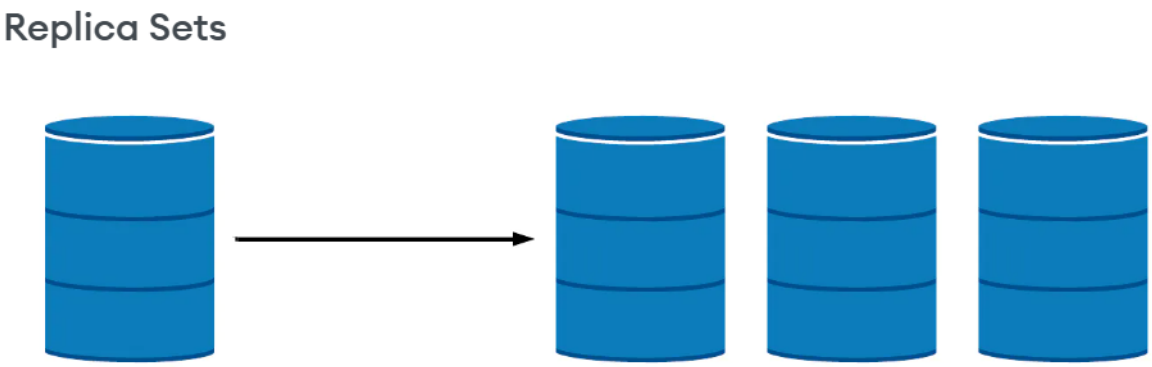

FIGURE 15 – La réplication permet une haute disponibilité, une gestion de la redondance et de la défaillance, ainsi qu'une réduction des goulots d'étranglement sur les opérations de lecture

7 Gestion du ou des caches mémoire (avec des schémas expliqués)

Il existe deux types de caches pour MongoDB qui sont gérés directement par le système.

7.1 Cache de lecture (Read Cache)

MongoDB utilise un cache de lecture appelé le cache de l'index (index cache) pour améliorer les performances des opérations de lecture. Lorsqu'une requête de lecture est exécutée, MongoDB vérifie d'abord si les données demandées sont présentes dans le cache de l'index. Si les données sont présentes, MongoDB les récupère directement à partir du cache, ce qui évite d'accéder aux fichiers de données sur le disque. Cela permet d'améliorer considérablement les performances en réduisant le temps d'accès aux données.

Le cache de l'index de MongoDB est basé sur une politique d'éviction LRU (Least Recently Used) qui remplace les entrées les moins récemment utilisées lorsque le cache atteint sa capacité maximale. La taille du cache de l'index est configurable et dépend de la quantité de mémoire disponible sur le système.

7.2 Cache d'écriture (Write Cache)

MongoDB utilise également un cache d'écriture appelé le journal des opérations (write-ahead log - WAL) pour optimiser les opérations d'écriture. Lorsqu'une requête d'écriture est exécutée, MongoDB stocke d'abord les opérations dans le journal des opérations avant de les appliquer aux fichiers de données sur le disque. Cela permet de regrouper plusieurs opérations d'écriture et de les traiter de manière plus efficace ultérieurement.

Le journal des opérations de MongoDB est conçu pour garantir la durabilité des données en cas de défaillance du système. Il assure que les opérations d'écriture sont enregistrées de manière durable avant d'être appliquées aux fichiers de données. Ainsi, même en cas de panne ou de redémarrage, les opérations d'écriture non encore traitées peuvent être récupérées à partir du journal des opérations pour restaurer l'intégrité des données.

8 Comparaison entre MongoDB et Cassandra

MongoDB et Cassandra sont deux bases de données NoSQL populaires qui diffèrent sur plusieurs critères importants, résumés ci-dessous :

8.1 Modèle de données

MongoDB utilise un modèle de données flexible basé sur des documents JSON, tandis que Cassandra utilise un modèle de colonnes basé sur des familles de colonnes. Cela signifie que MongoDB est plus adapté aux structures de données complexes et évolutives, tandis que Cassandra est optimisé pour les charges de travail nécessitant une évolutivité horizontale et une haute disponibilité.

8.2 Consistance

MongoDB offre une forte cohérence par défaut, garantissant que les lectures et les écritures reflètent immédiatement les dernières modifications. Cassandra, en revanche, offre une cohérence ajustable, permettant aux développeurs de choisir entre des niveaux de cohérence plus stricts ou plus souples en fonction de leurs besoins.

8.3 Partitionnement et réplication

Les deux bases de données prennent en charge le partitionnement et la réplication pour assurer la scalabilité et la disponibilité. Cependant, Cassandra utilise une approche de partitionnement distribué basée sur le hachage de clé, tandis que MongoDB utilise le partitionnement basé sur un index (sharding). Cela peut influencer les performances, la répartition des données et les capacités de requête.

8.4 Performances

Cassandra est réputée pour ses performances en écriture, grâce à son architecture distribuée optimisée pour une grande volumétrie d'écriture.

MongoDB excelle dans les opérations de lecture et d'écriture simples, mais peut être moins performant pour les charges de travail en écriture intensives.

8.5 Schéma

MongoDB est sans schéma, ce qui signifie qu'il permet une flexibilité maximale en termes de structure de données.

Cassandra, quant à elle, utilise un schéma défini au niveau de la famille de colonnes, nécessitant une modélisation plus rigide des données.

9 Sources

https://geekflare.com/fr/mongodb-sharding/
https://blog.dyma.fr/mongodb-la-base-de-donnees-nosql-la-plus-utilisee/
https://www.programmevitam.fr/ressources/Doc0.26.0/html/archi/archi-exploit-infra/services/mongodb.html
https://www.mongodb.com/docs/manual/indexes/#footnote-b-tree
https://www.cnblogs.com/zhaowenzhong/articles/5163896.html
https://datascientest.com/mongodb
https://experienceleague.adobe.com/docs/experience-manager-65/deploying/introduction/aem-with-mongodb.html?lang=fr
https://blog.servermania.com/mongodb-cluster/
https://welovedevs.com/fr/articles/mongodb-index/
https://www.mongodb.com/docs/manual/core/ranged-sharding/#:~:text=Range%2Dbased%20sharding%20involves%20dividing,documents%20within%20a%20contiguous%20range.
https://www.mongodb.com/docs/manual/core/hashed-sharding/
http://b3d.bdpedia.fr/replication.html
https://www.stackhero.io/fr-fr/services/MongoDB/documentations/Replica-set-HA
https://www.manageengine.com/fr/applications_manager/mongodb-monitoring.html
https://subscription.packtpub.com/book/web-development/9781783989126/1/ch01lvl1sec12/cassandra-s-architecture
https://hevodata.com/learn/cassandra-replication/
https://www.guru99.com/cassandra-architecture.html
https://levelup.gitconnected.com/cassandra-the-right-data-store-for-scalability-performance-a