UNIVERSITÉ
CÔTE D'AZUR

Rapport du projet Bases de Données

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

Rédigé par
Hadil AYARI
Nadjib KHAMMAR
Miléna KOSTOV
David PRIGODIN

15 juin 2023

Table des matières

Première partie
Cassandra

1 Modèles de données supportés

Cassandra est un système de gestion de base de données NoSQL qui prend en charge plusieurs modèles de données. Les principaux modèles pris en charge par Cassandra sont :

1.1 Modèle de colonnes

Cassandra organise les données en colonnes regroupées en familles de colonnes. Cela permet une flexibilité élevée dans la manière dont les données sont structurées et permet des requêtes rapides sur un sous-ensemble spécifique de colonnes.

1.2 Modèle de ligne

Cassandra prend également en charge un modèle de ligne traditionnel où les données sont organisées en lignes. Cependant, ce modèle n'est pas aussi performant que le modèle de colonnes pour les opérations de lecture et d'écriture massives.

1.3 Modèle de graphe

Cassandra propose également des fonctionnalités de modélisation de données de graphe grâce à son extension appelée Gremlin. Cela permet de stocker et de requêter des données de manière graphique, ce qui est particulièrement utile pour les applications orientées graphe.

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

La procédure d'installation du moteur Cassandra et des utilitaires nécessite quelques considérations particulières. Dans notre cas, nous avons opté pour l'utilisation d'une version plus ancienne, la version 3.11, qui est compatible avec Windows et est actuellement maintenue. Cela nous permet de simplifier le processus d'installation et de nous concentrer sur l'utilisation de Cassandra.

Pour commencer, il est important de noter que la dernière version de Cassandra, la version 4, n'est plus prise en charge sur Windows et est principalement destinée à être utilisée sur des systèmes Linux. Cependant, pour les utilisateurs de Windows souhaitant utiliser Cassandra V4, une option consiste à utiliser Docker pour exécuter l'environnement de développement.

Une fois que nous avons choisi d'utiliser la version 3.11 de Cassandra, nous devons également prendre en compte les versions correspondantes du JDK (Java Development Kit) et de Python dans nos variables d'environnement. Cela garantit que notre configuration est compatible avec les exigences de la version choisie.

Une fois l'installation terminée et la configuration vérifiée, nous pouvons accéder au répertoire des binaires Cassandra situé sous le répertoire C :/ et exécuter la commande cassandra.bat -f. Cette commande lancera le serveur Cassandra, qui sera alors prêt à fonctionner. Dans un autre terminal, nous pouvons utiliser la commande cqlsh pour démarrer l'interface en ligne de commande de Cassandra, également connue sous le nom de CQLSH.

CQLSH est l'outil que nous utilisons pour interagir avec Cassandra et sa base de données en utilisant le langage de requête CQL (Cassandra Query Language). C'est à travers CQLSH que nous pouvons démarrer des sessions, exécuter des requêtes, modifier des données et manipuler notre base de données.

3 Architecture du moteur NoSQL

L'architecture de Cassandra repose sur un modèle distribué et décentralisé. Elle est conçue pour fournir une haute disponibilité et une évolutivité horizontale. L'architecture de base de Cassandra comprend les éléments suivants :

3.1 Cluster

Un cluster Cassandra est composé de plusieurs nœuds (serveurs) qui travaillent ensemble pour stocker et gérer les données. Chaque nœud participe également à la réplication des données.

3.2 Nœud

Un nœud est une instance de Cassandra s'exécutant sur une machine physique ou virtuelle. Chaque nœud peut stocker une partie des données et effectue des opérations de lecture/écriture.

3.3 Clé de partition

Cassandra utilise une clé de partition pour déterminer comment les données sont distribuées dans le cluster. La clé de partition est utilisée pour calculer l'emplacement d'une donnée spécifique dans le cluster.

3.4 Réplication

Cassandra réplique les données sur plusieurs nœuds pour garantir la tolérance aux pannes et la disponibilité. Les données répliquées sont stockées sur des nœuds supplémentaires, appelés nœuds de réplication, pour assurer la redondance des données.

4 Méthode de partitionnement

Cassandra utilise une méthode de partitionnement basée sur le hachage pour répartir les données sur les nœuds du cluster. Le partitionnement est effectué en utilisant la clé de partition spécifiée pour chaque donnée. Le processus de partitionnement se déroule en plusieurs étapes :

4.1 Clé de partitionnement

Chaque donnée dans Cassandra est associée à une clé de partition, qui est utilisée pour déterminer l'emplacement de la donnée dans le cluster. La clé de partition est généralement choisie de manière à répartir uniformément les données sur l'ensemble du cluster.

4.2 Fonction de hachage

Cassandra utilise une fonction de hachage pour convertir la clé de partition en une valeur numérique qui est ensuite utilisée pour localiser le nœud responsable du stockage de la donnée.

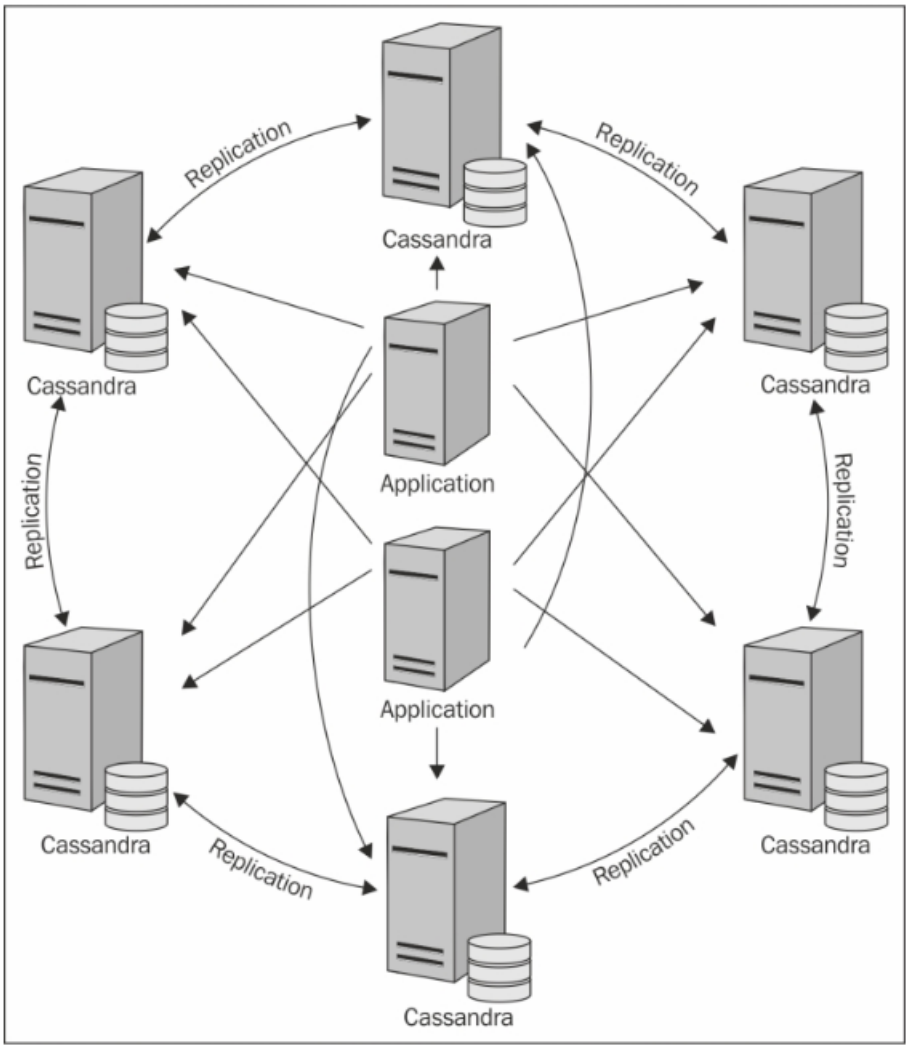

FIGURE 1 – Architecture Cassandra

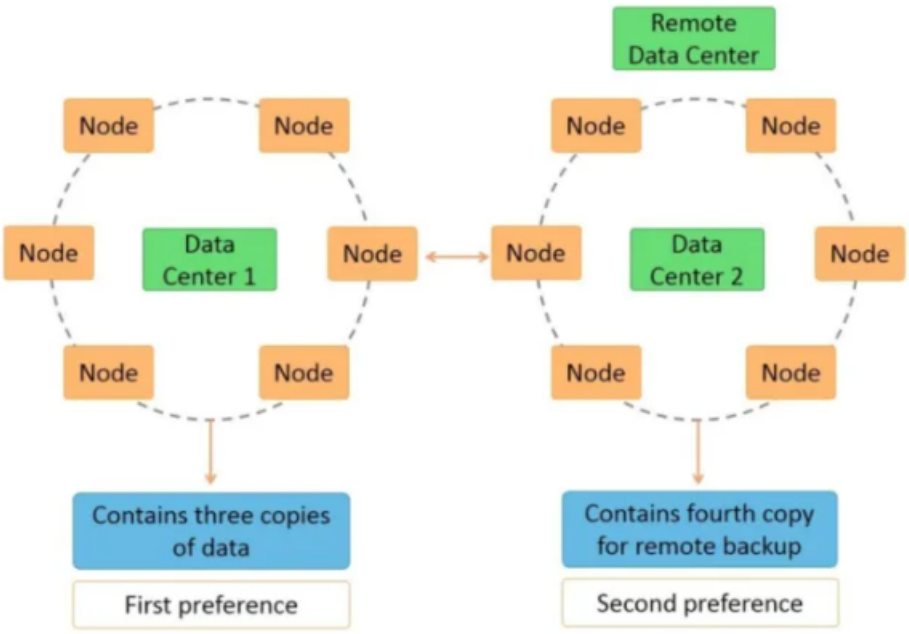

FIGURE 2 – Exemple d'une structure de réplication de Cassandra

4.3 Stratégie de partitionnement

Cassandra propose plusieurs stratégies de partitionnement, notamment la partitionnement uniforme, le partitionnement par plage et le partitionnement par jeton. Chaque stratégie a ses propres avantages et convient à des cas d'utilisation spécifiques.

5 Méthode de réplication

La réplication des données est un aspect essentiel de la conception de Cassandra. Elle garantit la disponibilité et la redondance des données en les copiant sur plusieurs nœuds.

L'architecture de réplication de Cassandra se compose généralement des composants suivants :

— Un nœud qui stocke les données.

— Un centre de données qui regroupe plusieurs nœuds répartis dans différents emplacements.

— Un cluster qui contient plusieurs centres de données.

— Une table de validation, une mémoire tampon en mémoire (mem-table), une table de stockage stable (SS table) et un filtre de Bloom pour prendre en charge le bon fonctionnement des composants internes de l'architecture.

Voici les étapes clés de la méthode de réplication dans Cassandra :

5.1 Facteur de réplication

Cassandra permet de définir le nombre de répliques de chaque donnée, appelé facteur de réplication. Le facteur de réplication détermine le nombre de copies de chaque donnée dans le

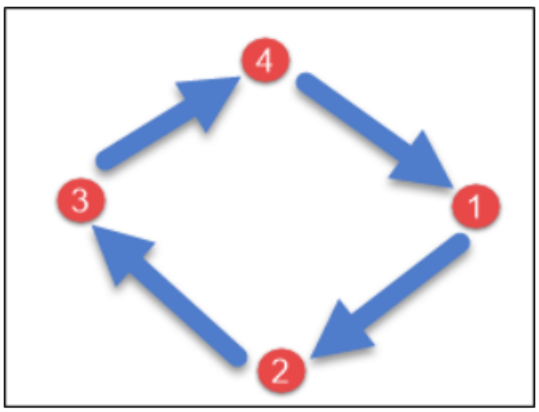

FIGURE 3 – Stratégie simple de réplication - cette stratégie est utilisée lorsqu'il n'y a qu'un seul centre de données. Elle place la première réplique sur le nœud sélectionné par le partitionneur. Ensuite, les répliques restantes sont placées dans le sens des aiguilles d'une montre sur l'anneau des nœuds.

cluster.

5.2 Stratégie de réplication

Cassandra offre différentes stratégies de réplication, telles que la réplication simple, la réplication en réseau, la réplication de centre de données et la réplication de plusieurs centres de données. Chaque stratégie permet de définir comment les données sont répliquées et distribuées sur les nœuds.

6 Montée en charge

Cassandra est conçu pour être hautement évolutif et permet une montée en charge linéaire en ajoutant simplement de nouveaux nœuds au cluster. La montée en charge dans Cassandra est réalisée par :

— **Ajout de nouveaux nœuds** Lorsque la charge de travail augmente, de nouveaux nœuds peuvent être ajoutés au cluster pour répartir la charge et augmenter la capacité de stockage et de traitement.

— **Rééquilibrage automatique** Cassandra effectue automatiquement le rééquilibrage des données lorsque de nouveaux nœuds sont ajoutés ou supprimés du cluster. Cela garantit que les données sont uniformément réparties sur l'ensemble du cluster.

— **Partitionnement efficace** Le partitionnement basé sur le hachage permet de répartir efficacement les données sur les nœuds du cluster, évitant ainsi les points chauds et permettant une distribution équilibrée de la charge.

Il y a deux types de scalabilité : horizontale et verticale.

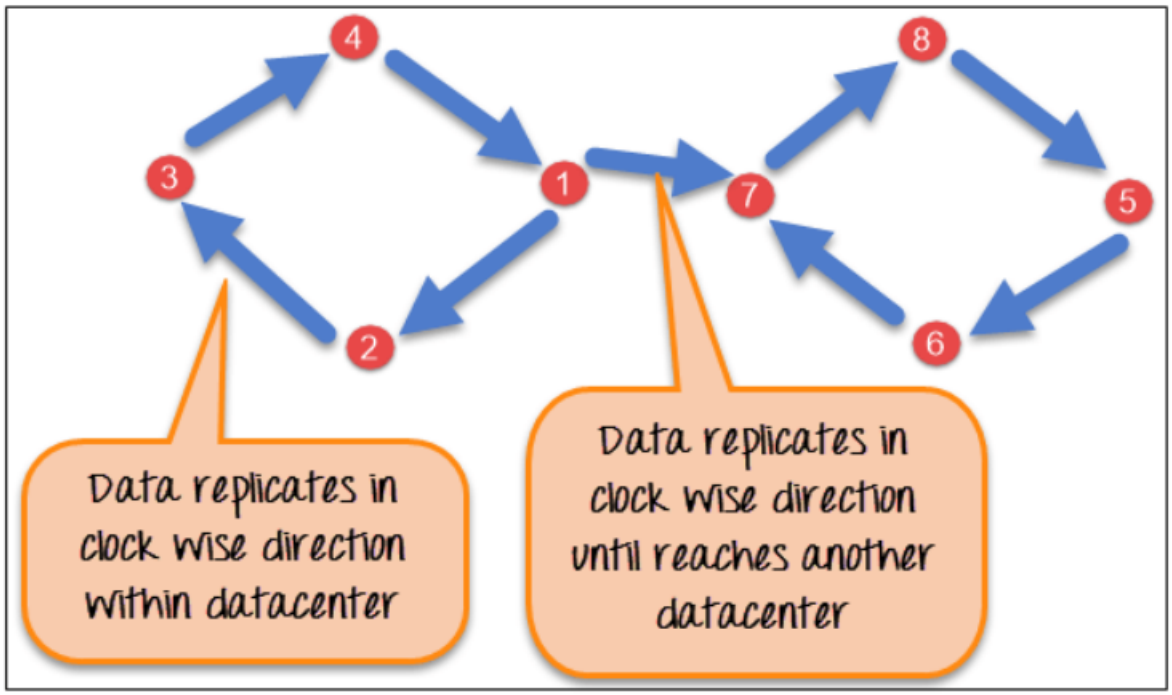

FIGURE 4 – Stratégie en réseau - cette stratégie est utilisée lorsqu'il y a plus de deux centres de données. Ici, les répliques sont configurées pour chaque centre de données séparément. Les répliques sont placées dans le sens des aiguilles d'une montre sur l'anneau jusqu'à ce qu'elles atteignent le premier nœud dans un autre rack. Cette stratégie tente de placer les répliques sur des racks différents dans le même centre de données.

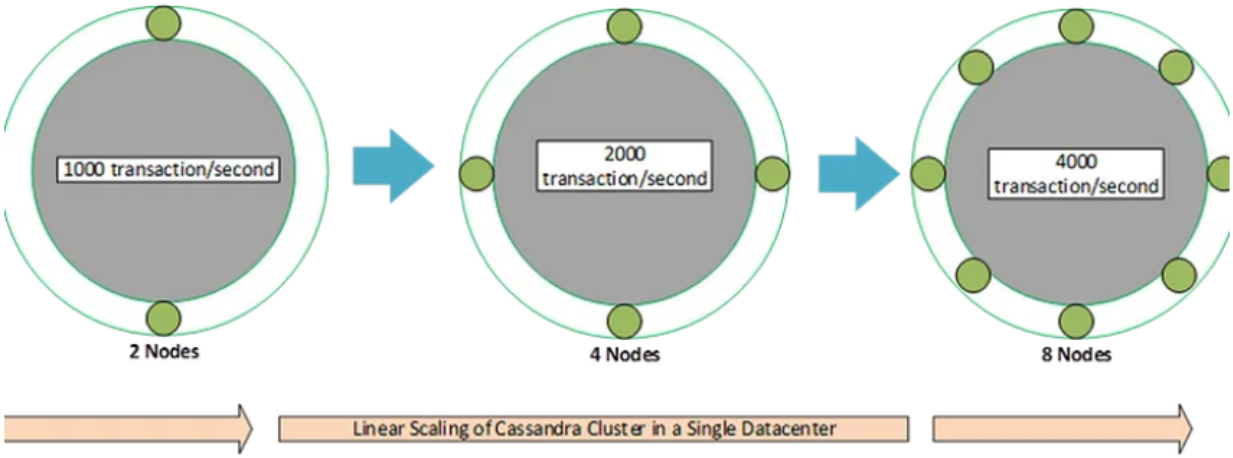

FIGURE 5 – Scalabilité verticale - Dans un seul datacenter - On peut augmenter la capacité du cluster en ajoutant plus de nœuds à un seul datacenter existant. La scalabilité verticale est généralement limitée par les capacités matérielles d'un datacenter spécifique.

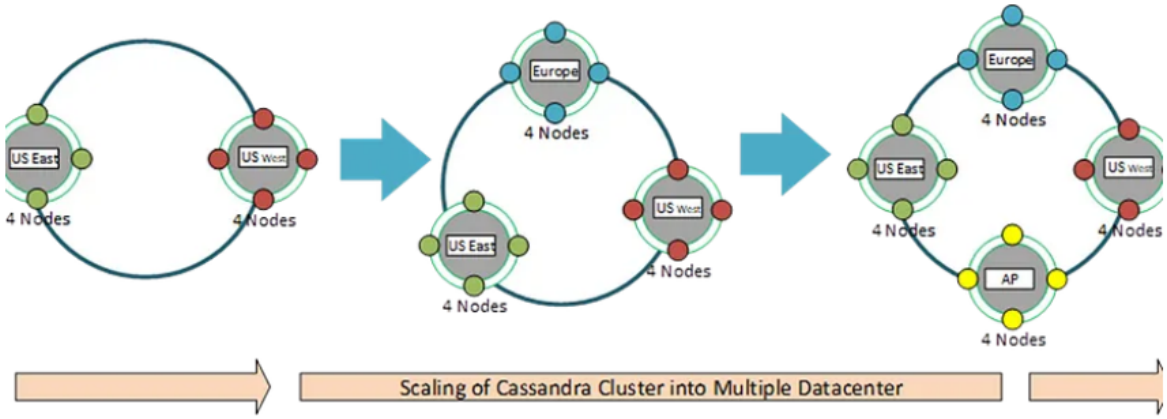

FIGURE 6 – Scalabilité horizontale - Plusieurs datacenters - On peut augmenter la capacité du cluster en ajoutant plus de datacenters, ce qui permet de distribuer la charge sur plusieurs sites géographiques. Cela permet d'améliorer la disponibilité des données et la tolérance aux pannes en répartissant les réplicas sur différents emplacements.

7 Gestion du ou des caches mémoire

Cassandra utilise un système de cache pour améliorer les performances des opérations de lecture. Les principaux caches utilisés par Cassandra sont les suivants :

7.1 Cache de clés (Key Cache)

Cassandra utilise un cache de clés pour stocker les résultats des requêtes de lecture les plus fréquentes. Ce cache permet d'éviter les accès disque coûteux en fournissant les données directement à partir de la mémoire.

7.2 Cache de blocs (Row Cache)

Cassandra propose également un cache de blocs pour stocker des lignes entières de données fréquemment accédées. Ce cache est utile lorsque des requêtes demandent un ensemble complet de données pour une clé de partition donnée.

Les caches de clés et de blocs peuvent être configurés et ajustés en fonction des besoins spécifiques de l'application pour optimiser les performances des opérations de lecture.

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://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