paul.millet@lameduse.fr

LaMeDuSe

SSA : Apache Cassandra, administration et exploitation

Version : 28/08/2024

Prérequis

- Connaissances de base de l'administration de Linux ou Windows.
- Connaissances de base d'une base de données relationnelle.

Objectifs

- Découvrir l'architecture du SGBD NoSQL Apache Cassandra
- Installer et configurer le SGBD NoSQL Apache Cassandra
- Créer une base de données et manipuler ses objets
- Découvrir les principes de l'optimisation et du développement

Tour de table

- Présentation des membres
- Vos attentes vis-à-vis de la formation

Programme de la formation

1. Présentation du SGBD Apache Cassandra
2. Installation et prise en main d'Apache Cassandra
3. Les bases de données sous Apache Cassandra
4. Interrogation des données sous Apache Cassandra
5. Administration et exploitation d'Apache Cassandra
6. Développement sous Apache Cassandra
7. Gestion des performances sous Apache Cassandra
8. Ressources des TP
9. Annexe & Ressources connexes

paul.millet@lameduse.fr

LaMeDuSe

SSA : Apache Cassandra, administration et exploitation

Présentation du SGBD Apache Cassandra

1. Apache Cassandra dans le monde du NoSQL.
2. Les cas d'utilisation d'Apache Cassandra.
3. Les éléments à prendre en compte pour utiliser Apache Cassandra.
4. Choix technique et architecture Apache Cassandra.

Présentation du SGBD Apache Cassandra

Qu'est-ce qu'Apache Cassandra ?

- Base de données NoSQL distribuée.
- Haute disponibilité et scalabilité linéaire.
- Conçue pour gérer de grandes quantités de données réparties sur plusieurs serveurs.
- Absence de Single Point of Failure (SPOF).

Historique et évolution :

- Créé par Facebook en 2008 pour gérer le moteur de recherche dans leur messagerie.
- Open-sourcé en 2009 et adopté par Apache Software Foundation.
- Large adoption par de nombreuses entreprises (Netflix, Twitter, Apple, etc.).

Caractéristiques Clés d'Apache Cassandra

Principales caractéristiques :

- Scalabilité linéaire : Ajout de nœuds sans perte de performance.
- Répartition et tolérance aux pannes : Réplication automatique des données sur plusieurs nœuds.
- Modèle de données flexible : Basé sur un modèle orienté colonne.
- Performance : Optimisé pour des opérations d'écriture intensives.
- Disponibilité continue : Pas de point unique de défaillance.

Cas d'utilisation adaptés :

- Applications nécessitant une haute disponibilité.
- Gestion de grands volumes de données.
- Systèmes distribués multi-régions/multi-data centers.

Apache Cassandra dans le monde du NoSQL

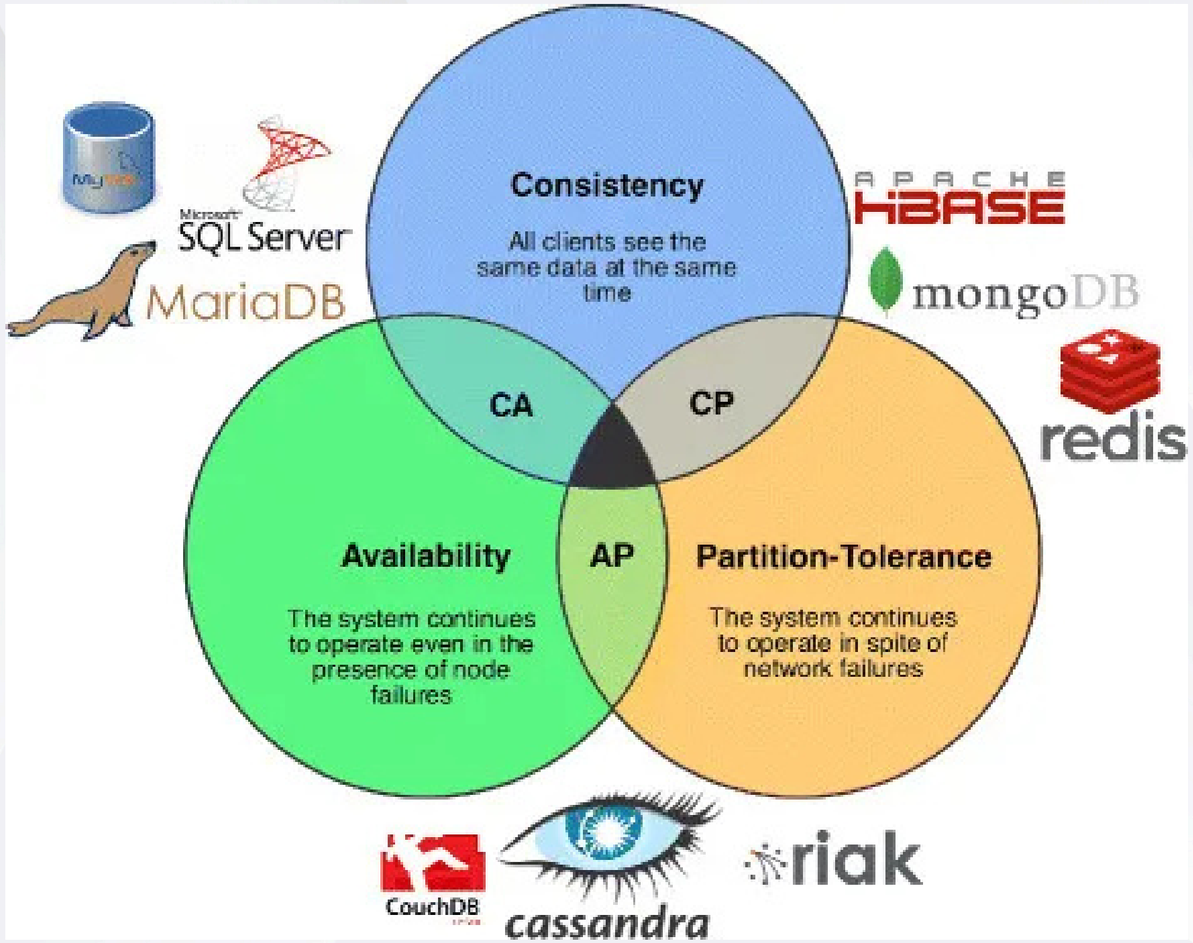
Vue d'ensemble des bases de données NoSQL :

- Types de bases de données NoSQL : clé-valeur, orientées document, graphe, orientées colonne.
- Positionnement d'Apache Cassandra : Base de données orientée colonne.

Comparaison avec d'autres solutions NoSQL :

- MongoDB : Orienté document, plus flexible pour les données semi-structurées.
- Couchbase : Orienté clé-valeur/document, avec intégration de fonctionnalités analytiques.
- HBase : Orienté colonne, intégré à l'écosystème Hadoop, mais moins flexible.

Apache Cassandra dans le monde du NoSQL



Apache Cassandra dans le monde du NoSQL

Pourquoi choisir Cassandra ?

- Haute disponibilité et résilience.
- Écritures intensives et lecture rapide avec des compromis cohérents.
- Scalabilité horizontale sans dégradation de performance.

Cas d'utilisation d'Apache Cassandra

Études de cas typiques :

- Applications de messagerie : Stockage et récupération rapide de messages.
- Suivi de l'activité utilisateur : Collecte de logs et données analytiques en temps réel.
- Gestion des capteurs IoT : Stockage des flux de données provenant de millions de capteurs.
- Recommandations personnalisées : Traitement et stockage de grandes quantités de données utilisateur.

Cas d'utilisation d'Apache Cassandra

Exemples d'implémentation :

- Netflix : Utilisé pour gérer les recommandations et la personnalisation de contenu.
- Uber : Utilisé pour la gestion de géolocalisation et des prix dynamiques.
- Discord: Message utilisateur et donnés de la plateforme

Éléments à prendre en compte pour utiliser Apache Cassandra

Considérations techniques :

- Compatibilité avec les besoins de l'application (écritures intensives, faible latence).
- Exigences en matière de cohérence et de tolérance aux pannes.
- Complexité de la gestion des clusters et du modèle de données.

Considérations business :

- Coûts d'infrastructure et d'exploitation.
- Support communautaire et professionnel.
- Adoption par les pairs et ressources disponibles (développeurs, consultants).

Choix technique et architecture d'Apache Cassandra

Composants clés de l'architecture Cassandra :

- Cluster : Ensemble de nœuds formant un système distribué.
- Nœud (Node) : Unité de base contenant des données.
- Ring : Topologie de base du cluster, chaque nœud est responsable d'une plage de données.
- Token : Identifiant unique assigné à un nœud pour répartir les données.

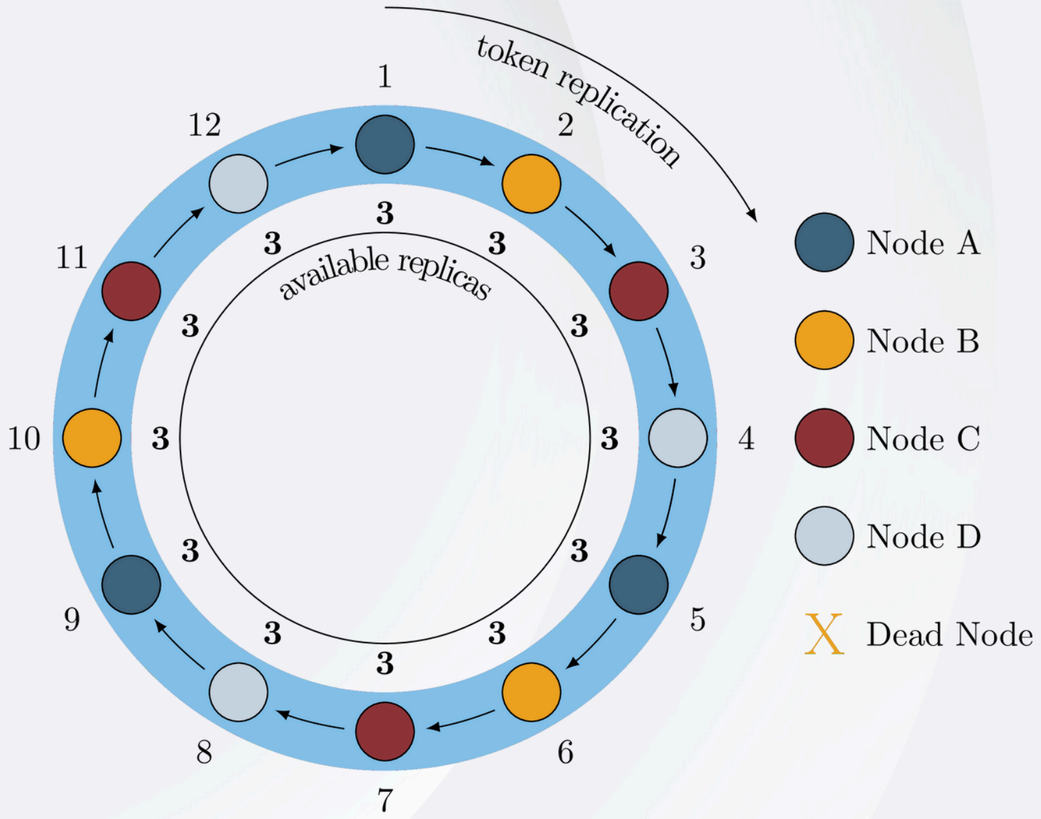
Modèles de déploiement :

- On-Premise : Déploiement interne pour plus de contrôle. (DataStax OpsCenter)
- Cloud : Flexibilité et scalabilité, exemple avec DataStax Astra.
- Hybride : Combinaison des deux pour une balance entre coût et performance.

Ring Architecture

Composants clés de l'architecture Cassandra :

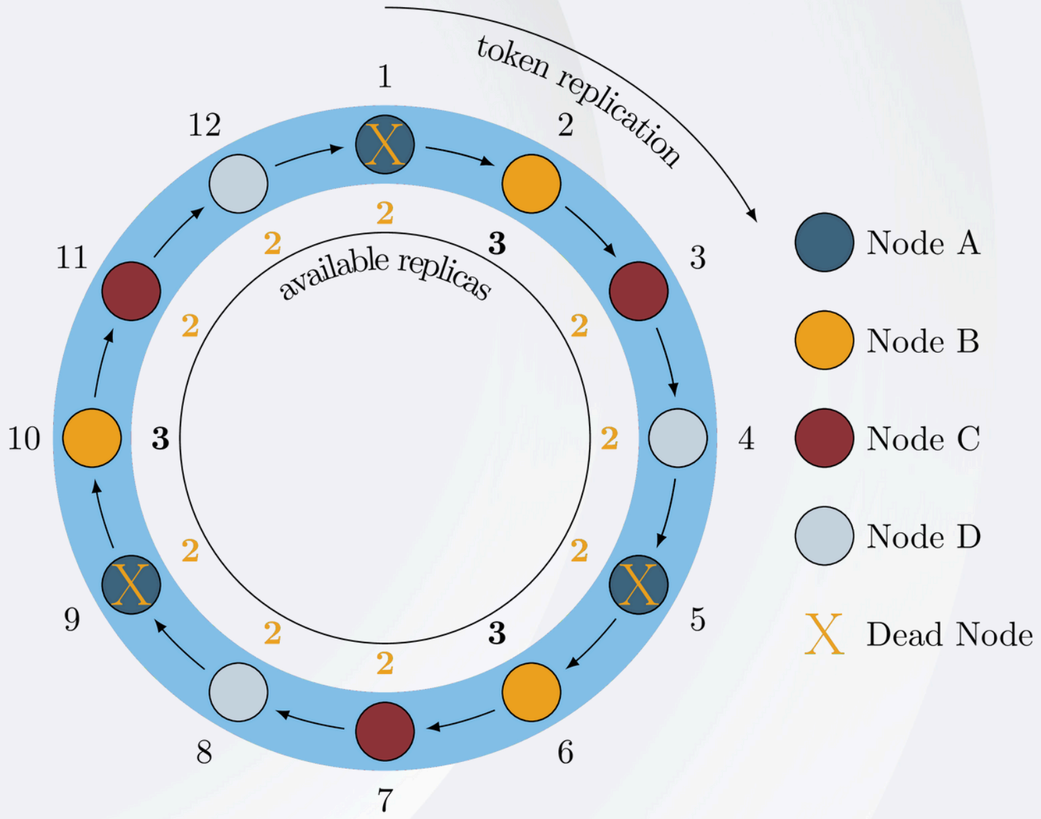
- Virtual Node (noeud virtuel) : utilisé pour la distribution de donnée



Ring Architecture

Composants clés de l'architecture Cassandra :

- Virtual Node (noeud virtuel) : utilisé pour la distribution de donnée



1.9-add2

Choix technique et architecture d'Apache Cassandra

Architecture de cluster optimale :

- Planification de la topologie du réseau.
- Choix des stratégies de réplication (SimpleStrategy, NetworkTopologyStrategy).
- Configuration pour la haute disponibilité et la tolérance aux pannes.

Mode	SimpleStrategy	NetworkTopologyStrategy
Utilisation typique	Développement/test	Production
Gestion des datacenters	❌ Non	✅ Oui
Gestion des racks	❌ Non	✅ Oui
Résilience	Faible	Élevée
Facilité de configuration	Très simple	Plus détaillée
Multi-datacenter	❌ Non	✅ Oui

Choix technique et architecture d'Apache Cassandra

```
CREATE KEYSPACE test_network
WITH replication = {
  'class': 'NetworkTopologyStrategy',
  'dc1': 3,
  'dc2': 2
};
```

```
CREATE KEYSPACE test_simple
WITH replication = {
  'class': 'SimpleStrategy',
  'replication_factor': 3
};
```

Récapitulatif

Recapitulatif des Points Clés :

- Apache Cassandra est une solution NoSQL adaptée aux environnements nécessitant une haute disponibilité et une scalabilité horizontale.
- Ses caractéristiques en font un choix privilégié pour des applications nécessitant une gestion intensive de données distribuées.
- Le choix d'Apache Cassandra doit être basé sur des besoins spécifiques de performance, de résilience et de capacité d'évolutivité.

Questions et Discussion :

- Questions ouvertes aux participants.
- Discussion sur les cas d'utilisation potentiels dans leur contexte d'entreprise.

paul.millet@lameduse.fr

LaMeDuSe

SSA : Apache Cassandra, administration et exploitation

Installation et prise en main d'Apache Cassandra

1. Les prérequis d'installation (sources, plateformes, etc.).
2. Configuration d'Apache Cassandra.
3. Mise en place d'une topologie du Cluster.
4. Validation de l'installation.

Prérequis d'Installation d'Apache Cassandra

Sources pour télécharger Apache Cassandra :

- Site officiel d’Apache Cassandra : cassandra.apache.org.
- Distributions disponibles (tar.gz, packages RPM, DEB).
- Version recommandée pour la formation.

Plateformes supportées :

- Linux : Red Hat, CentOS, Ubuntu, Debian.
- Windows : Support limité, recommandé pour le développement.
- macOS : Support via Homebrew.

Prérequis d'Installation d'Apache Cassandra

Configuration matérielle requise :

- Mémoire : Minimum 8 Go de RAM, recommandé 16 Go+.
- CPU : Multi-cœur, 2 GHz+.
- Stockage : Disques SSD recommandés pour la performance d'I/O.

Pré-requis logiciels :

- Java Runtime Environment (JRE) : Version 8 ou supérieure.
- Python : Version 2.7 pour le script d’installation.

Configuration d'Apache Cassandra

Fichiers de configuration principaux :

- cassandra.yaml : Fichier de configuration principal.
  - Définir les paramètres clés : cluster_name, seed_provider, data_file_directories, commitlog_directory.
  - Réglages de performance : memtable_flush_writers, compaction_throughput.
- cassandra-env.sh : Configuration des paramètres JVM.
  - Optimisation de la mémoire et des threads.
  - Passer des paramètres a la JVM

Sources :

- https://cassandra.apache.org/doc/4.1/cassandra/configuration/cass_yaml_file.html

Configuration d'Apache Cassandra

Sécurité et gestion des utilisateurs :

- Authentification et autorisation (PasswordAuthenticator, Role-Based Access Control).
  - Default : authenticator: AllowAllAuthenticator
- SSL/TLS pour la communication sécurisée entre les nœuds.

Configurer la surveillance et le monitoring :

- Activer JMX (Java Management Extensions) pour la supervision.
- Configurer le système de journaux pour les diagnostics.

Mise en Place d'une Topologie de Cluster

Architecture de Cluster Cassandra :

- Nœuds (Nodes) : Composants de base d'un cluster.
- Cluster : Groupe de nœuds connectés.
- Rings : Organisation logique des nœuds pour la répartition des données.
- Seed Nodes : Nœuds de référence pour l'initialisation du cluster.

Types de Nœuds et leur Rôle :

- Nœuds de données : Stockage et traitement des données.
- Seed Nodes : Faciliter la découverte et la communication entre les nœuds.

Mise en Place d'une Topologie de Cluster

Configuration réseau et topologie :

- Configurer listen_address et rpc_address pour les communications internes et externes.
- Utiliser des adresses IP statiques pour les nœuds dans un environnement de production.

Validation de l'Installation

Vérification des services Cassandra :

- Vérifier que le service Cassandra est démarré et en cours d'exécution.
- Utilisation de la commande nodetool status pour vérifier l'état du cluster.

Tests de connectivité et d'intégrité :

- Utilisation de cqlsh pour se connecter au cluster et exécuter des requêtes de base.
- Vérifier les logs de Cassandra pour détecter les erreurs et les avertissements.

Outils de diagnostic :

- Utiliser nodetool pour diagnostiquer et surveiller le cluster.
- Commandes courantes : nodetool info, nodetool status, nodetool ring.

CQL - Création et usage d’un keyspace

```
CREATE KEYSPACE myks WITH replication = {
  'class': 'SimpleStrategy', 'replication_factor': 1
};

USE myks;

DESCRIBE KEYSPACES;
```

CQL - Création d’une table

```
CREATE TABLE users (
  id UUID PRIMARY KEY,
  name text,
  email text
);

DESCRIBE TABLES;
DESCRIBE TABLE users;
```

CQL - Insertion d’une donnée dans la table

```
INSERT INTO users (id, name, email)
VALUES (uuid(), 'Alice', 'alice@example.com');

SELECT * FROM users;

UPDATE users SET name = 'Bob' WHERE id = <uuid>;

DELETE FROM users WHERE id = <uuid>;
```

TP: Installation et Test de Bon Fonctionnement

Étape par étape de l'installation d'Apache Cassandra :

- Télécharger et extraire la distribution Apache Cassandra.
- Configurer les paramètres de base dans cassandra.yaml.
- Démarrer le service Cassandra sur un nœud.

Vérifier l'installation :

- Connecter au cluster avec cqlsh.
- Créer un keyspace de test et une table.
- Insérer et interroger des données pour vérifier le bon fonctionnement.

Scénario de dépannage courant :

- Erreurs fréquentes lors de l'installation et comment les résoudre.
- Utiliser les logs de Cassandra pour diagnostiquer les problèmes.

TP (alt): Installation et Test de Bon Fonctionnement

Étape par étape de l'installation d'Apache Cassandra :

- Executer docker run --name cassandra cassandra

Vérifier l'installation :

- docker exec -it cassandra /bin/sh
- Executer cqlsh.
- Créer un keyspace de test et une table.
- Insérer et interroger des données pour vérifier le bon fonctionnement.

Scénario de dépannage courant :

- Erreurs fréquentes lors de l'installation et comment les résoudre.
- Utiliser les logs de Cassandra pour diagnostiquer les problèmes.

TP: Mise en Place d'un Cluster de Cassandra

Installation de Cassandra sur plusieurs nœuds :

- Configurer les adresses IP et les fichiers cassandra.yaml pour chaque nœud.
- Définir les nœuds seed et démarrer le cluster.

Tests de réplication et de distribution des données :

- Créer une table avec une stratégie de réplication définie.
- Insérer des données et observer la distribution sur les différents nœuds.

Gestion des erreurs et des pannes :

- Simuler la panne d'un nœud et observer la récupération du cluster.
- Utilisation de nodetool repair et nodetool cleanup.

Conclusion

Recapitulatif des Points Clés :

- Connaissance des prérequis pour l'installation de Cassandra.
- Maîtrise de la configuration de base et des fichiers essentiels.
- Mise en place d'une topologie de cluster fonctionnelle.
- Validation de l'installation et dépannage de base.

Questions et Discussion :

- Discussion ouverte sur les défis d'installation rencontrés.
- Échanges sur les meilleures pratiques d'installation et de configuration.

paul.millet@lameduse.fr

LaMeDuSe

SSA : Apache Cassandra, administration et exploitation

Les bases de données sous Apache Cassandra

1. Rappel sur les différents modèles de stockage du NoSQL.
2. Mise en place du modèle de données orienté colonne.
3. Les objets sous Apache Cassandra : Keyspace, tables, index secondaires.
4. Exemple concret de mise en œuvre.
5. TP: Création de bases de données et manipulation des objets créés.

Rappel sur les Modèles de Stockage NoSQL

Introduction aux différents modèles NoSQL :

- Clé-Valeur : Stockage de données sous forme de paires clé-valeur.
- Orienté Document : Documents JSON/BSON (MongoDB).
- Graphes : Relations entre entités (Neo4j).
- Orienté Colonne : Colonnes et familles de colonnes (Cassandra).

Pourquoi choisir un modèle orienté colonne ?

- Optimisé pour des lectures et écritures rapides.
- Adapté pour des analyses de données massives.
- Bonne gestion des schémas en constante évolution.

Le Modèle de Données Orienté Colonne d'Apache Cassandra

Structure de base du modèle orienté colonne :

- Keyspace : Equivalent d'une base de données.
- Table : Collection de colonnes.
- Colonnes : Contiennent les données.
- Partiton Key : Détermine sur quel nœud les données sont stockées.

Caractéristiques des modèles orientés colonne :

- Flexibilité dans le schéma : Ajout/suppression de colonnes à la volée.
- Données groupées par partition pour optimiser les lectures.
- Modèle optimisé pour les écritures fréquentes.

Les Objets Sous Apache Cassandra

Les Keyspaces :

- Représentation d'une base de données dans Cassandra.
- Paramètres clés : Stratégie de réplication (SimpleStrategy, NetworkTopologyStrategy), Facteur de réplication.
- Commande de création : CREATE KEYSPACE.

Les Tables :

- Ensemble de colonnes organisées par lignes.
- Utilisation des clés de partition et de clustering.
- Commande de création : CREATE TABLE.

Les Objets Sous Apache Cassandra

Les Index Secondaires :

- Amélioration des performances de requêtes sur des colonnes non primaires.
- Cas d'utilisation adaptés et limitations.
  - Exemple : La ville de résidence d’un utilisateur
  - Limitation : Perte de performance sur les grandes tables
- Commande de création : CREATE INDEX.

Les Types d'Objets Avancés :

- Types définis par l'utilisateur (UDT).
- Collections (listes, sets, maps).

Les Objets Sous Apache Cassandra

```
CREATE TYPE keyspace_name.type_name (
    field_name1 data_type,
    field_name2 data_type,
    ...
);
```

Les Objets Sous Apache Cassandra

```
-- Switch to your keyspace
USE my_keyspace;


-- Create the custom type
CREATE TYPE address (
    street text,
    city text,
    zip_code int
);
```

Les Objets Sous Apache Cassandra

```
CREATE TABLE users (
    id UUID PRIMARY KEY,
    name text,
    home_address address,
    work_address address
);
```

Les Objets Sous Apache Cassandra

```
INSERT INTO users (id, name, home_address, work_address)
VALUES (
    uuid(),
    'Alice',
    { street: '123 Main St', city: 'Paris', zip_code: 75001 },
    { street: '42 Rue de Biz', city: 'Lyon', zip_code: 69000 }
);
```

Les Objets Sous Apache Cassandra

Non Frozen UDT

```
UPDATE users
SET home_address.city = 'Marseille'
WHERE id = <user-id>;
```

Frozen UDT

```
UPDATE users
SET address = {city: 'Marseille'....}
WHERE name = 'alice';
```

Les Objets Sous Apache Cassandra

SELECT name, home_address.city FROM users;

Les Objets Sous Apache Cassandra

```
ALTER TYPE keyspace_name.udt_name ADD new_field_name data_type;

ALTER TYPE keyspace_name.udt_name RENAME old_field_name TO
new_field_name;
```

Les Objets Sous Apache Cassandra

- Retrait d’un champ impossible
- La modification d'un type d'un champ est impossible

Niveau de consistance

https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/dml/dmlConfigConsistency.html#Table1.WriteConsistencyLevels

Exemple Concret de Mise en Œuvre

Contexte : Application de gestion d'utilisateurs

- Exemple de structure de données :
  - Keyspace : user_management.
  - Table : users (id, nom, email, date_inscription).
  - Index Secondaire : Sur l'email pour faciliter les recherches.

Implémentation avec des commandes CQL :

- CREATE KEYSPACE user_management ...
- CREATE TABLE users ...
- CREATE INDEX ON users(email)

Optimisation des Requêtes :

- Utilisation des clés de partition pour équilibrer la charge.
- Index secondaire pour les recherches ponctuelles.

TP: Création de Bases de Données et Manipulation d'Objets

Exercice pratique : Créer un Keyspace et une Table

- Création d'un Keyspace inventory.
- Création d'une Table products (id, nom, catégorie, prix).

Manipuler les objets créés :

- Insérer des données dans la table products.
- Mettre à jour les informations de produit.
- Supprimer des enregistrements spécifiques.

Optimiser les Requêtes :

- Utilisation de clés de partition pour les recherches.
- Création et utilisation d'un index secondaire pour améliorer les performances de requête.

Bonnes Pratiques

Comprendre les principes de modélisation orientés colonne :

- Concevoir le modèle de données en fonction des requêtes.
- Utiliser des clés de partition pour répartir les données de manière uniforme.

Stratégies d'indexation efficaces :

- Utiliser des index secondaires avec parcimonie.
- Préférer des clés composites pour optimiser les recherches.

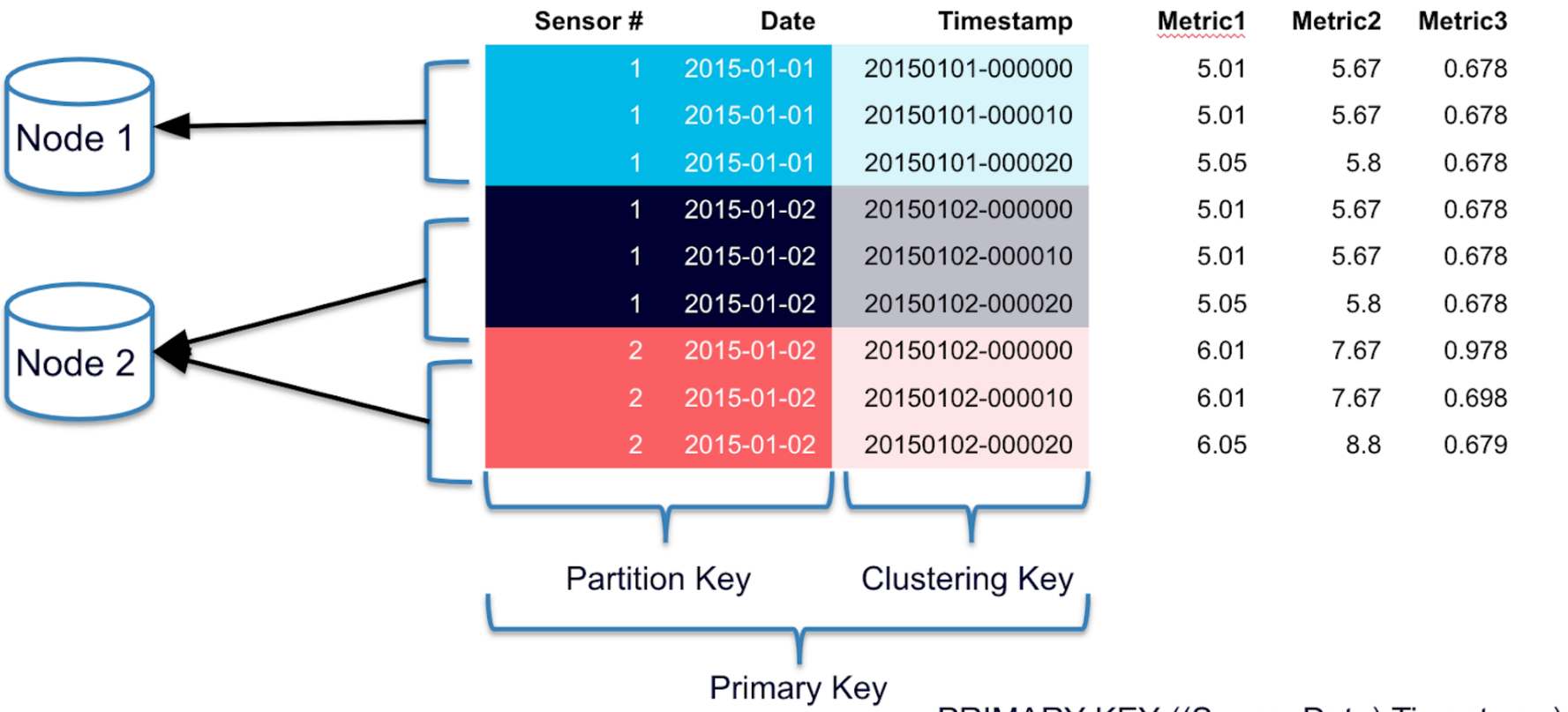
Gestion de la cohérence des données :

- Utilisation de BATCH pour des opérations transactionnelles.
- Choix de la stratégie de réplication et du facteur de réplication.

Performance et scalabilité :

- Privilégier des lectures légères et des écritures massives.
- Limiter la taille des partitions pour éviter les hotspots.

Bonnes Pratiques



Sensor #	Date	Timestamp	Metric1	Metric2	Metric3
1	2015-01-01	20150101-000000	5.01	5.67	0.678
1	2015-01-01	20150101-000010	5.01	5.67	0.678
1	2015-01-01	20150101-000020	5.05	5.8	0.678
1	2015-01-02	20150102-000000	5.01	5.67	0.678
1	2015-01-02	20150102-000010	5.01	5.67	0.678
1	2015-01-02	20150102-000020	5.05	5.8	0.678
2	2015-01-02	20150102-000000	6.01	7.67	0.978
2	2015-01-02	20150102-000010	6.01	7.67	0.698
2	2015-01-02	20150102-000020	6.05	8.8	0.679

PRIMARY KEY ((Sensor,Date),Timestamp)

Conclusion

Recapitulatif des Points Clés :

- Compréhension du modèle de données orienté colonne dans Cassandra.
- Maîtrise des objets clés : Keyspace, tables, index secondaires.
- Mise en pratique de la création et manipulation des objets.
- Meilleures pratiques pour la modélisation et optimisation des données.

Questions et Discussion :

- Échanges sur les défis de modélisation rencontrés.
- Discussion sur les cas d'utilisation spécifiques des participants.

4 - Interrogation des données sous Apache Cassandra

paul.millet@lameduse.fr

LaMeDuSe

SSA : Apache Cassandra, administration et exploitation

Interrogation des données sous Apache Cassandra

1. Utilisation du langage déclaratif CQL.
2. Prise en main du client interactif CQL pour réaliser les différentes opérations.
3. Accéder à Apache Cassandra via des API.
4. Les différentes opérations possibles sur les objets.

Introduction à CQL (Cassandra Query Language)

Qu'est-ce que CQL ?

- Langage déclaratif inspiré de SQL pour interroger et manipuler les données dans Apache Cassandra.
- Conçu pour les opérations de lecture/écriture optimisées dans un environnement distribué.

Principales caractéristiques de CQL :

- Syntaxe similaire à SQL pour la facilité d'apprentissage.
- Prise en charge des types de données spécifiques à Cassandra (blob, list, map, etc.).
- Optimisé pour les modèles de données orientés colonne.

Introduction à CQL (Cassandra Query Language)

Pourquoi utiliser CQL ?

- Permet des opérations de données complexes tout en garantissant la performance.
- Supporte les requêtes évolutives dans un environnement de Big Data.

Prise en Main du Client Interactif CQL (cqlsh)

Introduction à cqlsh :

- Outil en ligne de commande pour interagir avec Apache Cassandra.
- Permet l'exécution de commandes CQL, la création de schémas, et la manipulation de données.

Principales commandes cqlsh :

- Connexion au cluster : cqlsh <IP_adresse>
- Afficher les Keyspaces : DESCRIBE KEYSPACES;
- Créer un Keyspace : CREATE KEYSPACE ...;
- Afficher la structure d'une table : DESCRIBE TABLE <table>;

Prise en Main du Client Interactif CQL (cqlsh)

Travaux Pratiques :

- Connexion au cluster avec cqlsh.
- Création d'un keyspace et d'une table.
- Exécution de requêtes de base.

Accéder à Apache Cassandra via des API

Types d'API disponibles pour Apache Cassandra :

- CQL Native Driver API : Supporte plusieurs langages (Java, Python, Node.js, etc.).
- REST API : Utilisée pour des intégrations Web.
- Thrift API : Protocole original pour Cassandra, maintenant moins utilisé.

Utilisation des Drivers CQL :

- Java Driver : “com.datastax.oss.driver.api.core.CqlSession” pour la connexion et les requêtes.
- Python Driver : cassandra.cluster pour interagir avec le cluster.

Accéder à Apache Cassandra via des API

- Avantages de l'utilisation des API :
  - Intégration transparente avec des applications.
  - Gestion des opérations de lecture/écriture de manière asynchrone et performante.

Accéder à Apache Cassandra via des API

- Types d'opérations de base :
  - CRUD :
    - Create : INSERT INTO <table>...;
    - Read : SELECT * FROM <table>...;
    - Update : UPDATE <table> SET...;
    - Delete : DELETE FROM <table>...;
- Opérations Avancées :
  - Batch : Exécution de plusieurs requêtes dans une transaction légère.
    - BEGIN BATCH ... APPLY BATCH;
  - Lightweight Transactions (LWT) : Garanties de cohérence conditionnelle.
    - INSERT INTO ... IF NOT EXISTS;

Pratiques Recommandées pour les Requêtes CQL

- Concevoir des requêtes efficaces :
  - Limiter le nombre de partitions scannées par une requête.
  - Utiliser des clés de partition pour optimiser les recherches.
- Utiliser des filtres appropriés :
  - Préférer les filtres basés sur les clés de clustering et partition.
  - Éviter les requêtes full table scan.
- Gestion des index :
  - Utiliser des index secondaires avec précaution.
  - Éviter les index sur des colonnes à faible cardinalité.
- Optimisation des performances de requêtes :
  - Utiliser des requêtes préparées pour optimiser les performances.
  - Surveiller l'utilisation de la mémoire et des ressources CPU lors des requêtes complexes.

TP: Réalisation de Requêtes et Opérations

- Exercice pratique : Requêtes de base et avancées avec CQL :
  - Créer et manipuler des keyspaces et tables.
  - Effectuer des requêtes de sélection avec des filtres avancés.
  - Utiliser les Batch pour des opérations de groupe.
- Utiliser les API pour interroger les données :
  - Utiliser le driver d’un langage que vous connaissez pour exécuter des requêtes CQL.
  - Intégrer des opérations Cassandra dans une application Python.

Conclusion

- Recapitulatif des Points Clés :
  - Compréhension de l'utilisation de CQL pour interroger Apache Cassandra.
  - Maîtrise du client interactif cqlsh et des API pour accéder aux données.
  - Réalisation d'opérations de données basiques et avancées.
  - Application des meilleures pratiques pour l'optimisation des requêtes.
- Questions et Discussion :
  - Questions ouvertes sur les défis rencontrés lors de l'interrogation de données.
  - Discussion sur les scénarios spécifiques et les meilleures pratiques.

5 - Administration et exploitation d'Apache Cassandra

paul.millet@lameduse.fr

LaMeDuSe

SSA : Apache Cassandra, administration et exploitation

Administration et exploitation d'Apache Cassandra

1. Les différents outils d'administration (NodeTool).
2. Supervision et monitoring d'Apache Cassandra.
3. Surveillance du Cluster.
4. L'import et l'export des données.

Administration et exploitation d'Apache Cassandra

Principaux outils disponibles :

- NodeTool : Utilitaire en ligne de commande pour gérer et surveiller le cluster Cassandra.
- OpsCenter : Interface graphique de Datastax pour l'administration et la supervision.
- JMX (Java Management Extensions) : Interface de gestion et de surveillance pour JVM Cassandra.
- Cassandra Reaper : Outil de gestion de la réparation des nœuds dans le cluster.

Administration et exploitation d'Apache Cassandra

NodeTool : Les commandes essentielles :

- nodetool status : Affiche l'état des nœuds du cluster.
- nodetool repair : Effectue une réparation sur un ou plusieurs nœuds.
- nodetool cleanup : Supprime les données tombées en désuétude.
- nodetool compactionstats : Affiche les statistiques de compactage.

Supervision et Monitoring d'Apache Cassandra

Pourquoi surveiller Cassandra ?

- Assurer la disponibilité et les performances du cluster.
- Identifier rapidement les problèmes de réseau, de ressources ou de nœud.
- Optimiser la configuration pour les charges de travail actuelles.

Les métriques essentielles à surveiller :

- Métriques de santé des nœuds : Latence, utilisation du CPU/mémoire, espace disque disponible.
- Métriques de performance des requêtes : Temps de réponse, taux d'erreur, taux de requêtes par seconde.
- Métriques de stockage : Taux de lecture/écriture, volume de données, compactage.

Supervision et Monitoring d'Apache Cassandra

Outils de monitoring recommandés :

- Prometheus/Grafana : Collecte et visualisation des métriques.
- Nagios/Zabbix : Surveillance et alertes.
- Cassandra Exporter : Expose les métriques JMX à Prometheus.

Surveillance du Cluster Apache Cassandra

Surveiller l'état du cluster :

- Utilisation de nodetool status pour vérifier l'état de tous les nœuds.
- Comprendre l'affichage : UN (Up, Normal), UJ (Up, Joining), DN (Down, Normal), etc.

Surveillance des opérations de lecture/écriture :

- Vérifier les erreurs et les échecs de requêtes (WriteTimeout, ReadTimeout).
- Utilisation de nodetool cfstats pour des statistiques sur les tables.

Surveillance de la santé des nœuds :

- Analyser les logs de Cassandra (fichiers system.log, debug.log).
- Surveillance des latences avec nodetool tpstats.

Surveillance du Cluster Apache Cassandra

Gestion des anomalies :

- Identifier les signes de contention : latence élevée, échecs de requêtes.
- Réparer les données corrompues ou incohérentes avec nodetool repair.

Import et Export des Données

Méthodes d'import et export dans Apache Cassandra :

- COPY FROM / COPY TO : Import/Export de données via cqlsh en utilisant des fichiers CSV.
- sstableloader : Outil pour charger des fichiers SSTable directement dans Cassandra.

Apache Spark : Utilisation de Spark pour des opérations ETL avancées.

- Exemple de commande : Import/Export avec cqlsh
- COPY <table> FROM 'file.csv' WITH HEADER = TRUE;
- COPY <table> TO 'file.csv' WITH HEADER = TRUE;

Import et Export des Données

Bonnes pratiques d'import/export :

- Vérifier les formats de données pour éviter les erreurs de chargement.
- Utiliser sstableloader pour des chargements massifs sur des clusters distribués.
- Effectuer des exports réguliers pour la sauvegarde.

Pratiques Recommandées pour l'Administration de Cassandra

Optimisation des performances :

- Planifier des réparations régulières pour éviter les incohérences de données.
- Éviter les réparations globales fréquentes pour minimiser l'impact sur les performances.
- Utiliser le compactage manuel pour réduire la fragmentation.

Gestion des ressources :

- Surveiller l'utilisation du disque et ajuster la taille des SSTables.
- Limiter la taille des caches pour éviter l'utilisation excessive de la mémoire.
- Utiliser des systèmes de fichiers performants (ext4, xfs) pour les données.

Pratiques Recommandées pour l'Administration de Cassandra

Sauvegarde et restauration :

- Effectuer des snapshots réguliers pour la sauvegarde.
- Comprendre le processus de restauration de snapshots en cas de sinistre.

Travaux pratiques : Optimisation et Sauvegarde

- Planifier une tâche de réparation automatique.
- Configurer des snapshots réguliers pour la sauvegarde des données.

TP: Mise en œuvre de la Supervision et du Monitoring

Exercice pratique : Mise en place d'un système de monitoring

- Installer et configurer Prometheus et Grafana pour collecter les métriques.
- Créer un tableau de bord Grafana pour visualiser les performances du cluster.

Configurer des alertes basées sur les métriques :

- Définir des seuils critiques pour l'utilisation du CPU, la latence des requêtes, etc.
- Mettre en place des alertes par email ou webhook en cas de dépassement de seuil.

Analyse des logs et résolution de problèmes :

- Configurer l'export des logs Cassandra vers un système centralisé.
- Analyser les logs pour détecter et corriger les anomalies.

paul.millet@lameduse.fr

LaMeDuSe

SSA : Apache Cassandra, administration et exploitation

Développement sous Apache Cassandra

1. Comprendre et utiliser l'API Thrift.
2. Examiner une application Apache Cassandra.
3. Les objets sous Apache Cassandra : colonnes composites, listes ordonnées, rangs espacés, indices secondaires.
4. Gestion de la cohérence en lecture/écriture.
5. TP: Mise en œuvre de l'API Thrift.

Introduction à l'API Thrift

Qu'est-ce que l'API Thrift ?

- Protocole de communication utilisé initialement par Apache Cassandra pour permettre les interactions avec le cluster.
- Interface binaire qui fournit un moyen efficace de lire et écrire des données.
- NOTE : L’api Thrift a été supprimé dans la version 4.0 de cassandra

Caractéristiques de l'API Thrift :

- Bas niveau : Accès direct aux structures de stockage.
- Support pour plusieurs langages (Java, Python, etc.).
- Dépréciée en faveur de CQL (Cassandra Query Language) pour la plupart des opérations.

Introduction à l'API Thrift

Quand utiliser Thrift ?

- Cas d'utilisation nécessitant des opérations spécifiques au protocole bas niveau.
- Pour la rétrocompatibilité avec des systèmes existants.

Limites de l'API Thrift :

- Pas de prise en charge de fonctionnalités modernes (comme les Lightweight Transactions).
- Complexité accrue par rapport à l'utilisation de CQL.

Introduction à l'API Thrift

Exemple d'utilisation avec Java :

- Configuration du client Thrift pour se connecter à Cassandra.
- Utilisation des méthodes de l'API pour créer des keyspaces et des colonnes.

```
TTransport transport = new TSocket("127.0.0.1", 9160);
transport.open();
Cassandra.Client client = new Cassandra.Client(new TBinaryProtocol(transport));
client.set_keyspace("my_keyspace");
// Opérations CRUD avec Thrift
```

Objets Avancés sous Apache Cassandra

Colonnes Composites :

- Combinaison de plusieurs valeurs pour créer une clé de colonne.
- Utile pour les structures de données complexes.

Listes Ordonnées :

- Stockage d'une liste de valeurs avec maintien de l'ordre d'insertion.
- Utilisation de collections comme list, set, map.

Rangs Espacés (Wide Rows) :

- Utilisation de plusieurs milliers de colonnes dans une même ligne.
- Optimisé pour les scénarios nécessitant un grand nombre d'écritures séquentielles.

Colonnes composites

```
CREATE TABLE messages (
  user_id text,
  folder text,
  timestamp timestamp,
  message text,
  PRIMARY KEY ((user_id), folder, timestamp)
);
```

Objets Avancés sous Apache Cassandra

Indices Secondaires :

- Permettent d'effectuer des recherches sur des colonnes non partitionnées.
- À utiliser avec précaution pour éviter les dégradations de performances.

Gestion de la Cohérence en Lecture/Écriture

Les différents niveaux de cohérence :

- ANY : La requête est réussie si au moins un nœud confirme l'opération.
- ONE : La requête est réussie si un nœud de la réplique confirme l'opération.
- QUORUM : La majorité des répliques doivent confirmer l'opération.
- ALL : Tous les nœuds de la réplique doivent confirmer l'opération.

Choisir le bon niveau de cohérence :

- Lecture rapide, écriture rapide : Cohérence faible (ex : ONE).
- Cohérence stricte : Utilisation de QUORUM ou ALL.
- Impact sur la latence : Plus le niveau de cohérence est élevé, plus la latence est importante.

Gestion de la Cohérence en Lecture/Écriture

Stratégies de gestion de la cohérence :

- Utiliser Read Repair pour maintenir la cohérence des lectures.
- Utiliser des transactions légères pour garantir l'unicité des écritures.

Pratiques Recommandées

Modélisation de données :

- Conception du schéma basé sur les requêtes (Query-Driven Design).
- Minimiser les écritures croisées entre les clusters pour éviter les conflits de cohérence.

Optimisation des performances :

- Utiliser des requêtes préparées pour réduire le coût des requêtes répétitives.
- Éviter les requêtes complexes qui nécessitent un scan de toutes les partitions.

Pratiques Recommandées

Utilisation efficace des collections :

- Limiter la taille des collections (set, list, map) pour éviter une surcharge de mémoire.
- Préférer des modèles de données dénormalisés pour les lectures rapides.

Gestion de la réplication et de la résilience :

- Choisir la bonne stratégie de réplication (SimpleStrategy, NetworkTopologyStrategy).
- Configurer des points de terminaison de données pour une récupération rapide.

TP: Mise en œuvre de l'API Thrift et des Objets Avancés

Manipulation des Objets Avancés :

- Créer et gérer des colonnes composites et des listes ordonnées.
- Configurer des indices secondaires et tester leurs effets sur les requêtes.

Expérimenter avec la Cohérence :

- Configurer différents niveaux de cohérence pour les lectures et les écritures.
- Observer les effets sur les performances et la disponibilité des données.

LaMeDuSe

SSA : Apache Cassandra, administration et exploitation

Gestion des performances sous Apache Cassandra

1. L'indexation sous Apache Cassandra.
2. Architecture optimale pour Apache Cassandra.
3. TP: Mise en œuvre de l'indexation sous Apache Cassandra.

Introduction à l'Indexation sous Apache Cassandra

Pourquoi l'indexation est-elle importante ?

- Accélère les recherches sur des colonnes non partitionnées.
- Optimise les requêtes complexes qui nécessitent de filtrer par plusieurs critères.

Types d'index sous Apache Cassandra :

- Index Secondaire : Permet la recherche sur une colonne qui n'est pas la clé de partition.
- Index Matériel : Utilisation de tables matérialisées pour des vues indexées.
- Index Personnalisé : Index créé par l'utilisateur pour des besoins spécifiques.

Introduction à l'Indexation sous Apache Cassandra

Quand utiliser ou éviter les index :

- Utilisation recommandée pour des requêtes sur des ensembles de données restreints.
- À éviter pour des requêtes à grande échelle sur des colonnes à haute cardinalité (peut dégrader les performances).

Meilleures Pratiques pour l'Indexation sous Apache Cassandra

Créer des index efficaces :

- Choisir des colonnes avec une faible cardinalité pour les index secondaires.
- Utiliser les vues matérialisées pour les lectures rapides sur des données précises.

Limites et inconvénients des index secondaires :

- Augmentent les temps d'écriture car les index doivent être mis à jour à chaque insertion.
- Risques de saturation des ressources en cas d'utilisation abusive (fortes charges de lecture).

Meilleures Pratiques pour l'Indexation sous Apache Cassandra

Techniques alternatives à l'indexation :

- Modélisation des données basée sur les requêtes.
- Utiliser des tables dénormalisées pour éviter la nécessité d'index.

Optimisation de l'Architecture pour Apache Cassandra

Architecture Distribuée Optimale :

- Utiliser une configuration multi-datacenter pour garantir une haute disponibilité.
- Assurer une distribution équilibrée des données sur les nœuds pour éviter les "hotspots".

Configuration matérielle et logicielle :

- Choisir le matériel adapté (SSD, RAM élevée, CPU multi-core).
- Optimiser les configurations JVM pour réduire le "Garbage Collection" et augmenter la stabilité.

Optimisation de l'Architecture pour Apache Cassandra

Stratégies de configuration des paramètres :

- Ajuster la taille des mémoires caches (key_cache_size_in_mb, row_cache_size_in_mb).
- Configurer les niveaux de compression en fonction de la charge de travail.

Gestion des Performances de Lecture et Écriture

Optimisation des performances de lecture :

- Utiliser les caches de clés et de lignes pour des lectures rapides.
- Configurer le niveau de cohérence pour un équilibre entre performance et disponibilité.

Amélioration des performances d'écriture :

- Optimiser la fréquence des réparations pour éviter des conflits de données.
- Utiliser le compactage périodique pour réduire la fragmentation et améliorer l'efficacité des écritures.

Gestion des Performances de Lecture et Écriture

Gestion des Compactions :

- Choisir le bon algorithme de compactage (Size-Tiered, Leveled, Time-Window).
- Configurer les stratégies de compactage selon le type de charge de travail (OLTP, OLAP).

Surveillance et Tuning des Performances

Outils de surveillance des performances :

- NodeTool : Utiliser nodetool cfstats et nodetool tpstats pour surveiller les statistiques des tables et des threads.
- JMX (Java Management Extensions) : Surveillance des métriques de performance via des outils comme JConsole.

Techniques de tuning des performances :

- Analyser les métriques de latence et de taux d'erreur.
- Ajuster les paramètres de gc_grace_seconds pour les clusters avec des opérations de suppression fréquentes.

Surveillance et Tuning des Performances

Exemples de métriques à surveiller :

- Temps moyen de latence des lectures et écritures.
- Utilisation de la mémoire et des disques.
- Fréquence et durée des compactions.

Surveillance et Tuning des Performances

Exemples de métriques à surveiller :

- Temps moyen de latence des lectures et écritures.
- Utilisation de la mémoire et des disques.
- Fréquence et durée des compactions.

Outils

Cassandra Reaper

- Outil d’automatisation pour réparer la consistance d’un cluster cassandra et possède une webUI

DataStax OpsCenter

- Outil de gestion de cycle de vie de cluster Cassandra