Qu'es-ce que Cassandra ?

Créée par Facebook pour ses besoins de haute disponibilité, performance et gestion de gros volumes de données, Cassandra a été mis à disposition de la communauté open source depuis 2008 et reprise par la fondation Apache.

Cassandra est un système de gestion de base de données NoSQL conçu pour stoker de gros volumes de données de manière distribuée sur plusieurs serveurs.

Petit lexique

SGBD : Système de gestion de base de données.

NoSQL : C'est un type de SGBD non relationnel.

Haute disponibilité : Les données sont toujours accessibles.

Système distribué : Les données répliquées sur différents serveurs et dans différents datacenters.

Keyspace : C'est le schéma de base de données dans Cassandra, il comporte les tables et le modèle de réplication.

CSV : Format simple de fichier texte représentant des données séparées par des virgules, points virgules etc. … souvent utilisé pour les exports / imports de données.

CQL : Cassandra Query Language

A qui s'adresse ce guide ?

Ce mémento s'adresse aux utilisateurs de Cassandra comme aux personnes souhaitant avoir une vue succincte sur ses usages et bonnes pratiques.

Il permet une utilisation immédiate et concrète depuis l'installation jusqu'aux premières requêtes.

Opérations en CQL :

Pour commencer

Afficher les différents KEYSPACEs :
DESCRIBE KEYSPACEs;

Créer un KEYSPACE :
Attention, les stratégies de réplications sont difficiles à modifier en production, ce n'est pas sans impact.
Un seul Datacenter :
CREATE KEYSPACE villes WITH REPLICATION = {'class': 'SimpleStrategy' , 'replication_factor': 4};
Attention à ne pas choisir un replication_factor suppérieur au nombre de nœuds présents sur le datacenter.
Plusieurs Datacenters :
CREATE KEYSPACE villes WITH REPLICATION = {'class': 'NetworkTopologyStrategy', 'Paris': 4, 'Toulouse': 3, 'Lille': 2} ;
Attention à ne pas choisir un replication_factor suppérieur au nombre de nœuds présent sur les datacenters.

Créer une table :
CREATE TABLE villes.coordonnees (nom text, code_postal text, latitude text, longitude text, nom_affiche text, primary key (nom, code_postal));
Ici la clef est composée des deux champs « nom » et « code postal »
Dans le cas présent les données ayant la même clef seront stockées sur le même nœud.

Afficher les propriétés d'une table :
DESCRIBE villes.coordonnees ;

Insérer des données :
INSERT into villes.coordonnees (nom, code_postal, latitude, longitude, nom_affiche) values ('Clumanc', '04330', '6.416670', '1.25', 'clumanc');

Supprimer une colonne :
ALTER TABLE villes.coordonnees drop colonne_a_supprimer;

Sélectionner des données :
SELECT * FROM villes.coordonnees;

Importer des données depuis un CSV :
COPY villes.coordonnees FROM 'liste_villes.csv' WITH DELIMITER = ';';

Exporter des données vers un CSV :
COPY villes.coordonnees to 'liste_villes.csv' WITH DELIMITER = ';';

Requêtes simples

Sélectionner des données

Sélectionner toutes les données :
SELECT * FROM villes.coordonnees;

Sélectionner des données et limiter le nombre de lignes retournées :
SELECT * FROM villes.coordonnees limit 10;

Sélectionner des données selon la clef primaire :
SELECT * FROM villes.coordonnees WHERE nom = 'Lagny-sur-Marne' and code_postal = '77400' ;

Sélectionne des données selon un seul élément de la clef primaire :
SELECT * FROM villes.coordonnees WHERE code_postal = '77400' ALLOW FILTERING ;
Attention cette requête peut affecter les performances de la base de données, c'est pour cette raison qu'il faut « forcer » la requête avec ALLOW FILTERING
Une bonne pratique consiste à utiliser LIMIT pour éviter la perte de performances :
*SELECT * FROM villes.coordonnees WHERE code_postal = '77400' LIMIT 5 ALLOW FILTERING ;*

Compte le nombre de lignes :
select count(*) from villes.coordonnees;
Attention cette requête peut affecter les performances de la base de données, elle doit interroger tous les nœuds.

Supprimer des données :
delete from villes.coordonnees where nom = 'Dampmart';
Possible que sur une clef pour des raisons de performances.

Mettre à jour des données

Mettre à jour un champ :
UPDATE villes.coordonnees set nb_habitants = 20718 WHERE nom = 'Lagny-sur-Marne' and code_postal = '77400';

Opérations sur les tables

Ajouter une colonne :
ALTER TABLE villes.coordonnees ADD nb_habitants int;

Supprimer une colonne :
ALTER TABLE villes.coordonnees DROP nb_habitants;

Vider une table :
TRUNCATE villes.coordonnees ;

Architecture distribuée schéma

Opérations sur le KEYSPACE

Vérifier les stratégies de réplication et coefficient de réplication :
SELECT * FROM system_schema.keyspaces ;

Changer le coefficient de réplication :
ALTER KEYSPACE villes WITH replication = {'class': 'SimpleStrategy' , 'replication_factor': 5} ;

Créer des vues

Nous pouvons créer une vue depuis une requête pour augmenter les performances :
CREATE MATERIALIZED VIEW villes.grandes_villes as SELECT * from villes.coordonnees WHERE nom in ('Paris', 'Marseille') AND nom IS NOT NULL AND code_postal IS NOT NULL PRIMARY KEY (nom, code_postal);
La requête select doit vérifier que les clefs ne sont pas nulles et la nouvelle clef doit etre egale ou plus restrictive que la table d'origine.

On peut désormais faire la requete suivante :
SELECT * FROM villes.grandes_villes ;
Cette requête est optimale au niveau performances;

Opérations sur les utilisateurs

Créer un utilisateur :

CREATE USER bda WITH PASSWORD 'bdaPassword' SUPERUSER;

CREATE USER simple_user WITH PASSWORD 'userPassword' NOSUPERUSER;

Supprimer un utilisateur :

DROP USER simple_user

Lister les utilisateurs :

LIST USERS

Comment Cassandra stocke et accède aux données.

- Ecriture dans le commit log
- Ecriture des données dans la memtable
- Ecriture des données depuis la memtable vers la sstable
  - Automatique lorsque la memtable va etre pleine
  - Peut être déclenché manuellement (opération d'admin)

La lecture des données suit un workflow dépendant de différents niveaux de cache.
Le fichier cassandra.yaml permet de configurer ces différents niveaux :

Row cache	row_cache_size_in_mb row_cache_save_period
Partition key cache	key_cache_size_in_mb

Configuration de cassandra.yaml

Le fichier est situé dans un emplacement différent selon le type d'installation :
Installation par package : /etc/cassandra/cassandra.yaml
Installation par décompression : <dossier d'installation>/cassandra/
conf/cassandra.yaml

Données à modifier :

cluster_name: 'MonCluster'
Nom du cluster

authenticator: PasswordAuthenticator
Permet de se connecter en tant que cassandra pour créer les autres utilisateurs

data_file_directories:
- /opt/cassandra/data
Permet de définir le dossier des sauvegardes

commitlog_directory: /opt/cassandra/logs/commitlog
Permet de définir où seront les logs / ne doit pas être sur la même partition que les datas

listen_address : Ip de la machine

- seeds: "10.9.8.7, 20.19.18.17"
Permet de définir les seeds pour démarrage des discussions entres nœuds

Principales utilisations de nodetool

nodetool	Affiche les différentes commandes possibles
nodetool status	Affiche l'état des nœuds connectés notamment : Datacenter, Mémoire utilisée, données contenues, Rack
nodetool tpstats	Affiche l'état des taches du nœud courant.
nodetool netstats	Affiche l'état du reseau du nœud courant.
nodetool tablestats <table>	Affiche les information des tables si pas d'argument, sinon affiche les informations de la table précisée
nodetool repair	Reconstruit les index
nodetool flush	Flush les données de memtable vers sstable Nécessaire avant de faire des Backups
nodetool decommission	Permet de dé-commissionner un nœud, l'option -h permet de dé-commissionner un autre nœud.
nodetool snapshot <keyspace>	Permet de faire une sauvegarde du KEYSPACE. Pour restaurer les données : • Stopper le nœud. • Vider la table • Recopier le contenu du snapshot dans le dossier des datas

Pour résumer

La communication entre les nœuds est assurée par **GOSSIP,** les problèmes matériels ou réseaux sont repérés par les **SNITCH** présents sur les nœuds.

Les données sont réparties et répliquées sur les **Nœuds** selon les stratégies de réplications définies au niveau des **Keyspaces**, elles peuvent être SimpleStrategy si sur un seul **Datacenter** et **NetworkTopologyStrategy** si sur plusieurs **Datacenters,** cette fonction de répartition est assurée par les **Partitioners.**

Attention : Les erreurs les plus communes dans l'utilisation de Cassandra sont principalement d'utiliser Cassandra comme une base de données relationnelle.

Exemples :

- Pour stocker des données cohérentes: le temps de propagation peut fausser les résultats (prix d'un article, calcul d'un panier).
- Faire des opérations obligeant le système à interroger tous les nœuds.

Rédaction :
Yann-Eric Devars