eBook Gratuit

APPRENEZ cassandra

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#cassandra

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: cassandra

It is an unofficial and free cassandra ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official cassandra.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec Cassandra

Remarques

La base de données Apache Cassandra est le bon choix lorsque vous avez besoin d'évolutivité et de haute disponibilité sans compromettre les performances. L'évolutivité linéaire et la tolérance aux pannes éprouvée sur le matériel ou l'infrastructure cloud, en font la plate-forme idéale pour les données critiques. Le support de Cassandra pour la réplication sur plusieurs centres de données est le meilleur de sa catégorie, offrant une latence plus faible pour vos utilisateurs et la tranquillité d'esprit de savoir que vous pouvez survivre aux pannes régionales.

ÉPROUVÉ

Cassandra est utilisée par Constant Contact, CERN, Comcast, eBay, GitHub, GoDaddy, Hulu, Instagram, Intuit, Netflix, Reddit, The Weather Channel et plus de 1500 autres entreprises qui disposent de grands ensembles de données actifs.

TOLÉRANCE DE PANNE

Les données sont automatiquement répliquées sur plusieurs nœuds pour la tolérance aux pannes. La réplication entre plusieurs centres de données est prise en charge. Les nœuds en échec peuvent être remplacés sans interruption.

PERFORMANT

Cassandra dépasse systématiquement les alternatives NoSQL les plus répandues dans les tests de performances et les applications réelles, principalement en raison des choix architecturaux fondamentaux.

DÉCENTRALISÉ

Il n'y a pas de points de défaillance uniques. Il n'y a pas de goulots d'étranglement sur le réseau. Chaque nœud du cluster est identique.

SCALABLE

Parmi les plus importants déploiements de production, notons Apple, avec plus de 75 000 nœuds stockant plus de 10 Go de données, Netflix (2 500 nœuds, 420 To, plus de 1 000 milliards de requêtes par jour), le moteur de recherche chinois Easou (270 nœuds, 300 To, plus de 800 millions reqests par jour) et eBay (plus de 100 nœuds, 250 To).

DURABLE

Cassandra convient aux applications qui ne peuvent pas se permettre de perdre des données, même en cas de panne d'un centre de données entier.

VOUS ÊTES EN CONTRÔLE

Choisissez entre une réplication synchrone ou asynchrone pour chaque mise à jour. Les opérations asynchrones hautement disponibles sont optimisées avec des fonctionnalités telles que le transfert interposé et la réparation en lecture.

ÉLASTIQUE

Le débit de lecture et d'écriture augmente de manière linéaire à mesure que de nouvelles machines sont ajoutées, sans interruption ni interruption des applications.

Soutenu professionnellement

Les contrats et services de support de Cassandra sont disponibles auprès de tiers.

Versions

Version	Date de sortie
1.1.12	2013-11-19
1.1.9	2013-02-11
1.2.12	2013-11-28
1.2.13	2013-12-19
1.2.15	2014-02-19
1.2.16	2014-04-22
1.2.17	2014-06-25
1.2.18	2014-07-04
1.2.19	2014-11-14
1.2.6	2013-07-02
1.2.8	2013-07-27
2.0.10	2014-08-12
2.0.11	2014-10-17
2.0.12	2015-01-14
2.0.13	2015-03-20
2.0.14	2015-04-02
2.0.15	2015-06-01

Version	Date de sortie
2.0.16	2015-07-08
2.0.17	2015-09-18
2.0.5	2014-02-13
2.0.6	2014-04-02
2.0.7	2014-04-24
2.0.8	2014-06-13
2.0.9	2014-07-22
2.1.11	2015-10-12
2.1.12	2015-10-22
2.1.2	2014-11-20
2.1.3	2015-03-03
2.1.4	2015-04-02
2.1.5	2015-03-31
2.1.6	2015-06-09
2.1.7	2015-06-18
2.1.8	2015-07-03
2.1.9	2015-09-03
2.2.0	2015-05-14
2.2.0-beta1	2015-05-19
2.2.0-rc1	2015-06-04
2.2.0-rc2	2015-06-30
2.2.1	2015-08-25
2.2.2	2015-09-25
2.2.3	2015-10-12
2.2.4	2015-12-02

Version	Date de sortie
3.0.0	2015-01-26
3.0.0-alpha	2015-07-29
3.0.0-alpha1	2015-07-18
3.0.0-beta1	2015-07-10
3.0.0-beta2	2015-09-04
3.0.0-rc1	2015-07-16
3.0.0-rc2	2015-10-16
3.0.1	2015-12-04
3.0.2	2016-01-21
3.0.3	2015-11-24
3,0.4	2016-02-05
3.0.5	2016-04-02
3.0.6	2016-03-31
3.0.7	2016-05-24
3.0.8	2016-05-25
3.2.819	2016-01-05
3.4.950	2016-03-08
3.6.1076	2016-05-02
3.8.1199	2016-06-27
3.10.3004	2016-08-10

(J'ai obtenu ceci en utilisant un peu awk: `git log --tags --simplify-by-decoration --pretty="format:%ai %d" |egrep "\(tag: [0-9]"| awk -F" " '{ print $1 " " $5}'|awk -F"." '{print $1 "." $2 "." $3}'| awk -F" " '{print $2 " |" $1}'| sed 's/)//'|sed 's/,//'| sort -n|sort -u -t" " -k1,1 | awk '{print "|" $0 "|"}'`)

Examples

Installation ou configuration

Installation d'un nœud unique

1. Pré-installer NodeJS, Python et Java
2. Sélectionnez votre document d'installation en fonction de votre plate-forme http://docs.datastax.com/en/cassandra/3.x/cassandra/install/installTOC.html
3. Téléchargez les binaires Cassandra depuis http://cassandra.apache.org/download/
4. Décompressez le fichier téléchargé dans `<installation location>`
5. Lancez Cassandra en utilisant `<installation location>/bin/cassandra` OU démarrez Cassandra en tant que service - `[sudo] service cassandra start`
6. Vérifiez si Cassandra est opérationnel avec le `<installation location>/bin/nodetool status` .

Ex:

1. Dans l'environnement Windows, exécutez le fichier `cassandra.bat` pour démarrer le serveur Cassandra et `cqlsh.bat` pour ouvrir le terminal client CQL afin d'exécuter des commandes CQL.

L'installation d'un **seul nœud** peut être effectuée de deux manières.

Vous devriez avoir Oracle Java 8 ou OpenJDk 8 (préféré pour les versions de Cassandra> 3.0)

1. Installer un paquet Debian (installe Cassandra en tant que service)

Ajoutez la version de Cassandra au référentiel (remplacez le 22x par votre propre version, par exemple pour 27x utilisation)

```
echo "deb-src http://www.apache.org/dist/cassandra/debian 22x main" | sudo tee -a
/etc/apt/sources.list.d/cassandra.sources.list
# Update the repository
sudo apt-get update
# Then install it
sudo apt-get install cassandra cassandra-tools
```

Maintenant, Cassandra peut être démarré et arrêté en utilisant:

```
sudo service cassandra start
sudo service cassandra stop
```

Vérifiez le statut en utilisant:

```
nodetool status
```

Les répertoires de journaux et de données sont respectivement `/var/log/cassandra` et `/var/lib/cassandra` .

2. Installer une version de Cassandra sous forme d'archive binaire (installe Cassandra en tant que processus autonome)

Téléchargez la version Datastax:

```
curl -L  http://downloads.datastax.com/community/dsc-cassandra-version_number-bin.tar.gz | tar xz
```

Ou l'archive binaire Apache Cassandra manuellement (à partir du site http://www.apache.org/dist/cassandra/)

Maintenant, lancez ceci:

```
tar -xvzf dsc-cassandra-version_number-bin.tar.gz
```

Changez le répertoire pour installer l'emplacement:

```
cd install_location
```

Commencez Cassandra en utilisant:

```
sudo sh ./bin/cassandra
```

Arrête d'utiliser:

```
sudo kill -9 pid
```

Vérifier:

```
./bin/nodetool status
```

Et alto, vous avez un cluster de test à nœud unique pour Cassandra. Il suffit donc d'utiliser `cqlsh` dans le terminal pour le shell Cassandra.

La configuration de Cassandra peut être faite dans le dossier `cassandra.yaml` dans `conf` dans `install_location` .

Installation multi-nœuds

Installation en cluster multi-cc

Lire Démarrer avec Cassandra en ligne: https://riptutorial.com/fr/cassandra/topic/1682/demarrer-avec-cassandra

Chapitre 2: Cassandra - PHP

Examples

Application console simple

Téléchargez le pilote PHP Datastax pour Apache Cassandra à partir du site du projet Git et suivez les instructions d'installation.

Nous utiliserons le tableau **"users"** de l'application **KillrVideo** et du pilote PHP Datastax. Une fois que Cassandra est opérationnel, créez l'espace de clés et la table suivants:

```
//Create the keyspace
    CREATE KEYSPACE killrvideo WITH REPLICATION =
    { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };

//Use the keyspace
USE killrvideo;

// Users keyed by id
CREATE TABLE users (
    userid uuid,
    firstname text,
    lastname text,
    email text,
    created_date timestamp,
    PRIMARY KEY (userid)
);
```

Créez un fichier PHP avec votre éditeur préféré. Tout d'abord, nous devons nous connecter à notre **espace** clavier et cluster **«killrvideo»** :

```
build();
$keyspace  = 'killrvideo';
$session  = $cluster->connect($keyspace);
```

Insérons un utilisateur dans la table **"users"** :

```
execute(new Cassandra\SimpleStatement(
    "INSERT INTO users (userid, created_date, email, firstname, lastname)
      VALUES (14c532ac-f5ae-479a-9d0a-36604732e01d, '2013-01-01 00:00:00',
      'luke@example.com','Luke','Tillman')"
  ));
```

En utilisant le pilote PHP Datastax Cassandra, nous pouvons interroger l'utilisateur par userid:

```
execute(new Cassandra\SimpleStatement
          ("SELECT firstname, lastname, email FROM killrvideo.users
            WHERE userid=14c532ac-f5ae-479a-9d0a-36604732e01d"));

foreach ($result as $row) {
```

```
    printf("user: \"%s\" \"%s\" email: \"%s\" \n", $row['firstname'],
    $row['lastname'], $row['email']);
  }
```

Pour qu'un utilisateur mette à jour son adresse e-mail dans le système:

```
execute(new Cassandra\SimpleStatement
          ("UPDATE users SET email = 'language_evangelist@example.com'
            WHERE userid = 14c532ac-f5ae-479a-9d0a-36604732e01d"));

execute(new Cassandra\SimpleStatement
          ("SELECT firstname, lastname, email FROM killrvideo.users
            WHERE userid=14c532ac-f5ae-479a-9d0a-36604732e01d"));

foreach ($result as $row) {
    printf("user: \"%s\" \"%s\" email: \"%s\" \n", $row['firstname'],
    $row['lastname'], $row['email']);
  }
```

Supprimez l'utilisateur de la table et affichez toutes les lignes. Vous remarquerez que les informations de l'utilisateur ne reviennent plus après avoir été supprimées:

```
execute(new Cassandra\SimpleStatement
          ("DELETE FROM users WHERE userid = 14c532ac-f5ae-479a-9d0a-36604732e01d"));

execute(new Cassandra\SimpleStatement
          ("SELECT firstname, lastname, email FROM killrvideo.users
            WHERE userid=14c532ac-f5ae-479a-9d0a-36604732e01d"));

foreach ($result as $row) {
    printf("user: \"%s\" \"%s\" email: \"%s\" \n", $row['firstname'],
    $row['lastname'], $row['email']);
  }
```

La sortie devrait ressembler à ceci:

```
user: "Luke" "Tillman" email: "luke@example.com"
user: "Luke" "Tillman" email: "language_evangelist@example.com"
```

Références :

Premiers pas avec Apache Cassandra et PHP , DataStax Academy

Lire Cassandra - PHP en ligne: https://riptutorial.com/fr/cassandra/topic/2866/cassandra---php

Chapitre 3: Cassandra en tant que service

Introduction

Cette rubrique décrit comment démarrer Apache Cassandra en tant que service dans les plates-formes Windows et Linux. Rappelez-vous que vous démarrez également Cassandra à partir du répertoire bin en exécutant le script batch ou shell.

Examples

les fenêtres

1. Téléchargez le dernier démon apache commons à partir des distributions du projet Apache Commons .
2. Extrayez le démon commons dans **<répertoire d'installation de Cassandra> \ bin** .
3. Renommez le dossier extrait en démon.
4. Ajoutez **<répertoire d'installation de Cassandra>** comme **CASSANDRA_HOME** dans la variable d'environnement Windows.
5. Editez le fichier **cassandra.yaml** dans **<répertoire d'installation de Cassandra> \ conf** et supprimez le commentaire du répertoire data_file_directory, commitlog_directory, saved_cache_directory et définissez les chemins absolus.
6. Editez **cassandra.bat** dans **<répertoire d'installation de Cassandra> \ bin** et remplacez la valeur de PATH_PRUNSRV comme suit:

```
for 32 bit windows, set PATH_PRUNSRV=%CASSANDRA_HOME%\bin\daemon\
for 64 bit windows, set PATH_PRUNSRV=%CASSANDRA_HOME%\bin\daemon\amd64\
```

7. Modifiez **cassandra.bat** et configurez SERVICE_JVM pour le nom de service requis.

```
SERVICE_JVM="cassandra"
```

8. Avec les privilèges d'administrateur, exécutez l'installation de cassandra.bat à partir de l'invite de commandes.

Linux

1. Créez le script de démarrage /etc/init.d/cassandra.
2. Editez le contenu du fichier:

```
#!/bin/sh
```

```
#
# chkconfig: - 80 45
# description: Starts and stops Cassandra
# update daemon path to point to the cassandra executable
DAEMON=<Cassandra installed directory>/bin/cassandra
start() {
        echo -n "Starting Cassandra... "
        $DAEMON -p /var/run/cassandra.pid
        echo "OK"
        return 0
}
stop() {
        echo -n "Stopping Cassandra... "
        kill $(cat /var/run/cassandra.pid)
        echo "OK"
        return 0
}
case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  restart)
        stop
        start
        ;;
  *)
        echo $"Usage: $0 {start|stop|restart}"
        exit 1
esac
exit $?
```

3. Rendre le fichier exécutable:

   sudo chmod + x /etc/init.d/cassandra

4. Ajoutez le nouveau service à la liste:

   sudo chkconfig --add cassandra

5. Vous pouvez maintenant gérer le service à partir de la ligne de commande:

```
sudo /etc/init.d/cassandra start
sudo /etc/init.d/cassandra stop
sudo /etc/init.d/cassandra restart
```

Lire Cassandra en tant que service en ligne:
https://riptutorial.com/fr/cassandra/topic/10544/cassandra-en-tant-que-service

Chapitre 4: Clés Cassandra

Examples

Clé de partition, clé de regroupement, clé primaire

Cassandra utilise deux types de clés:

- les **clés de partition** sont responsables de la distribution des données entre les nœuds
- la **Clustering Key** est responsable du tri des données dans une partition

Une **clé primaire** est une combinaison de ceux-ci aux types. Le vocabulaire dépend de la combinaison:

- **clé primaire simple** : seule la clé de partition, composée d'une colonne
- **clé de partition composite** : seule la clé de partition, composée de plusieurs colonnes
- **clé primaire composée** : une clé de partition avec une ou plusieurs clés de clustering.
- **clé primaire composite et composée** : une clé de partition composée de plusieurs colonnes et de plusieurs clés de clustering.

La syntaxe PRIMARY KEY

Déclarer une clé

L'instruction de création de table doit contenir une expression `PRIMARY KEY` . La façon dont vous le déclarez est très importante. En un mot:

```
PRIMARY KEY(partition key)
PRIMARY KEY(partition key, clustering key)
```

Les parenthèses supplémentaires regroupent plusieurs champs dans une clé de partition composite ou déclarent une clé composite composée.

Exemples

Clé primaire **simple** :

```
PRIMARY KEY (key)
```

`key` est appelée la **clé de partition** .

(pour une clé primaire simple, il est également possible de mettre l'expression `PRIMARY KEY` après le champ, par exemple la `key int PRIMARY KEY`, par exemple).

Clé primaire **composée** :

```
PRIMARY KEY (key_part_1, key_part_2)
```

Contrairement à SQL, cela ne crée pas exactement une clé primaire composite. Au lieu de cela, il déclare `key_part_1` comme **clé de partition** et `key_part_2` comme **clé de cluster** . Tout autre champ sera également considéré comme faisant partie de la clé de regroupement.

Clés primaires **composites + composées** :

```
PRIMARY KEY ((part_key_1, ..., part_key_n), (clust_key_1, ..., clust_key_n))
```

La première parenthèse définit la **clé de partition composée** , les autres colonnes sont les clés de cluster.

Résumé de la syntaxe

- `(part_key)`
- `(part_key, clust_key)`
- `(part_key, clust_key_1, clust_key_2)`
- `(part_key, (clust_key_1, clust_key_2))`
- `((part_key_1, part_key_2), clust_key)`
- `((part_key_1, part_key_2), (clust_key_1, clust_key_2))`

Commande de clés et requêtes autorisées

La **clé de partition** est le **spécificateur minimum** requis pour exécuter une requête à l'aide d'une clause where.

Si vous déclarez une **clé de clustering composite** , l'ordre compte.

Disons que vous avez la clé primaire suivante:

```
PRIMARY KEY((part_key1, part_key_2), (clust_key_1, clust_key_2, clust_key_3))
```

Ensuite, les *seules requêtes valides* utilisent les champs suivants dans la clause `where` :

- `part_key_1` , `part_key_2`
- `part_key_1` , `part_key_2` , `clust_key_1`
- `part_key_1` , `part_key_2` , `clust_key_1` , `clust_key_2`
- `part_key_1` , `part_key_2` , `clust_key_1` , `clust_key_2` , `clust_key_3`

Exemple de requêtes non valides:

- `part_key_1` , `part_key_2` , `clust_key_2`
- Tout ce qui ne contient pas à la fois `part_key_1` , `part_key_2`
- ...

Si vous souhaitez utiliser `clust_key_2` , vous devez également spécifier `clust_key_1` , etc.

Ainsi, l'ordre dans lequel vous déclarez vos clés de cluster aura un impact sur le type de requêtes

que vous pouvez effectuer. Au contraire, l'ordre des champs de clé de la partition n'est pas important, car vous devez toujours tous les spécifier dans une requête.

Lire Clés Cassandra en ligne: https://riptutorial.com/fr/cassandra/topic/9071/cles-cassandra

Chapitre 5: Connexion à Cassandra

Remarques

Le pilote Cassandra de Datastax ressemble beaucoup au pilote Java JDBC MySQL.

`Session` , `Statement` , `PreparedStatement` sont présents dans les deux pilotes.

La connexion Singleton provient de cette question et réponse:
http://stackoverflow.com/a/24691456/671896

Côté fonctionnalité, Cassandra 2 et 3 sont identiques. Cassandra 3 a introduit une réécriture complète du système de stockage de données.

Examples

Java: inclure le pilote Cassandra DSE

Dans votre projet Maven, ajoutez ce qui suit à votre fichier `pom.xml` . Les versions suivantes sont pour Cassandra 3.x.

```
    <dependency>
        <groupId>com.datastax.cassandra</groupId>
        <artifactId>cassandra-driver-core</artifactId>
        <version>3.1.0</version>
    </dependency>
    <dependency>
        <groupId>com.datastax.cassandra</groupId>
        <artifactId>cassandra-driver-mapping</artifactId>
        <version>3.1.0</version>
    </dependency>
    <dependency>
        <groupId>com.datastax.cassandra</groupId>
        <artifactId>cassandra-driver-extras</artifactId>
        <version>3.1.0</version>
    </dependency>
    <dependency>
        <groupId>com.datastax.cassandra</groupId>
        <artifactId>dse-driver</artifactId>
        <version>1.1.0</version>
    </dependency>
```

Java: se connecter à une instance Cassandra locale

La connexion à Cassandra est très similaire à la connexion à d'autres sources de données. Avec Cassandra, les informations d'identification ne sont pas requises.

```
String cassandraIPAddress = "127.0.0.1";
String cassandraKeyspace = "myKeyspace";
String username = "foo";
```

```
String password = "bar";

com.datastax.driver.core.Cluster cluster = Cluster.builder()
    .addContactPoint(cassandraIPAddress)
    .withCredentials(username, password) // If you have setup a username and password for your
node.
    .build();

com.datastax.driver.core.Session session = cluster.connect(cassandraKeyspace);

com.datastax.driver.core.Metadata metadata = cluster.getMetadata();

// Output Cassandra connection status
System.out.println("Connected to Cassandra cluster: " + metadata.getClusterName() + " with
Partitioner: " + metadata.getPartitioner());

// Loop through your entire Cluster.
for (Host host : metadata.getAllHosts()) {
    System.out.println("Cassandra Host Address: " + host.getAddress() + " | Is Up = " +
host.isUp());
}
```

Java: Connecter en utilisant un singleton

```
public enum Cassandra {

    DB;

    private Session session;
    private Cluster cluster;
    private static final Logger LOGGER = LoggerFactory.getLogger(Cassandra.class);

    /**
     * Connect to the cassandra database based on the connection configuration provided.
     * Multiple call to this method will have no effects if a connection is already
established
     * @param conf the configuration for the connection
     */
    public void connect(ConnectionCfg conf) {
        if (cluster == null && session == null) {
            cluster =
Cluster.builder().withPort(conf.getPort()).withCredentials(conf.getUsername(),
conf.getPassword()).addContactPoints(conf.getSeeds()).build();
            session = cluster.connect(conf.getKeyspace());
        }
        Metadata metadata = cluster.getMetadata();
        LOGGER.info("Connected to cluster: " + metadata.getClusterName() + " with partitioner:
" + metadata.getPartitioner());
        metadata.getAllHosts().stream().forEach((host) -> {
            LOGGER.info("Cassandra datacenter: " + host.getDatacenter() + " | address: " +
host.getAddress() + " | rack: " + host.getRack());
        });
    }

    /**
     * Invalidate and close the session and connection to the cassandra database
     */
    public void shutdown() {
        LOGGER.info("Shutting down the whole cassandra cluster");
```

```
        if (null != session) {
            session.close();
        }
        if (null != cluster) {
            cluster.close();
        }
    }

    public Session getSession() {
        if (session == null) {
            throw new IllegalStateException("No connection initialized");
        }
        return session;
    }
}
```

Utiliser la connexion Singleton

```
public void cassandra() throws Exception {
    Cassandra.DB.connect();
    Cassandra.DB.getSession().execute(/* CQL | Statement | PreparedStatement */)
    Cassandra.DB.close();
}
```

Lire Connexion à Cassandra en ligne: https://riptutorial.com/fr/cassandra/topic/7845/connexion-a-cassandra

Chapitre 6: Réparations à Cassandra

Paramètres

Option / Drapeau	La description
option	*description de l'option*
-h	nom d'hôte. La valeur par défaut est "localhost". Si vous ne spécifiez pas qu'une réparation d'hôte est exécutée sur le même hôte que celui à partir duquel la commande est exécutée.
-p	Port JMX. La valeur par défaut est 7199.
-u	Nom d'utilisateur. Seulement requis si la sécurité JMX est activée.
-pw	mot de passe. Seulement requis si la sécurité JMX est activée.
drapeau	*description du drapeau*
-local	Ne compare et ne diffuse que les données provenant des nœuds du centre de données "local".
-pr	"Partitioner Range" réparer. Ne réparez que la plage de jetons principale pour un réplica. Plus rapide que la réparation de toutes les plages de vos répliques, car cela évite de réparer plusieurs fois les mêmes données. Notez que si vous utilisez cette option pour réparer un nœud, vous devez également l'utiliser pour le reste de votre cluster.
-par	Exécutez les réparations en parallèle. Obtient les réparations plus rapidement, mais limite considérablement la capacité du cluster à gérer les demandes.
-hôtes	Vous permet de spécifier une liste de noeuds délimités par des virgules à partir de laquelle vous souhaitez diffuser vos données. Utile si vous avez des nœuds connus pour être "bons". Bien qu'il soit documenté comme une option valide pour Cassandra 2.1+, il fonctionne également avec Cassandra 2.0.

Remarques

Cassandra Anti-Entropy Repairs:

La réparation anti-entropie chez Cassandra comporte deux phases distinctes. Pour réussir des réparations performantes, il est important de comprendre les deux.

- **Calculs de l'arborescence Merkle** : Ceci calcule les différences entre les noeuds et leurs

répliques.

- **Transmission de données** : en fonction des résultats des calculs de Merkle Tree, les données doivent être transmises d'un nœud à un autre. Ceci est une tentative de synchronisation des données entre les répliques.

Arrêter une réparation :

Vous pouvez arrêter une réparation en émettant une commande STOP VALIDATION à partir de nodetool:

```
$ nodetool stop validation
```

Comment savoir quand la réparation est terminée?

Vous pouvez vérifier la première phase de réparation (calculs Merkle Tree) en vérifiant les `nodetool compactionstats` .

Vous pouvez vérifier les flux de réparation à l'aide de `nodetool netstats` . Les flux de réparation seront également visibles dans vos journaux. Vous pouvez `grep` pour eux dans vos journaux système comme celui - ci:

```
$ grep Entropy system.log

INFO [AntiEntropyStage:1] 2016-07-25 07:32:47,077 RepairSession.java (line 164) [repair
#70c35af0-526e-11e6-8646-8102d8573519] Received merkle tree for test_users from /192.168.14.3
INFO [AntiEntropyStage:1] 2016-07-25 07:32:47,081 RepairSession.java (line 164) [repair
#70c35af0-526e-11e6-8646-8102d8573519] Received merkle tree for test_users from /192.168.16.5
INFO [AntiEntropyStage:1] 2016-07-25 07:32:47,091 RepairSession.java (line 221) [repair
#70c35af0-526e-11e6-8646-8102d8573519] test_users is fully synced
INFO [AntiEntropySessions:4] 2016-07-25 07:32:47,091 RepairSession.java (line 282) [repair
#70c35af0-526e-11e6-8646-8102d8573519] session completed successfully
```

Les flux de réparation actifs peuvent également être surveillés avec cette commande (Bash):

```
$ while true; do date; diff <(nodetool -h 192.168.0.1 netstats) <(sleep 5 && nodetool -h
192.168.0.1 netstats); done
```

ref: comment puis-je savoir si la réparation de nodetool est terminée

Comment rechercher les flux de réparation bloqués ou orphelins?

Sur chaque nœud, vous pouvez surveiller cela avec `nodetool tpstats` et vérifier tout ce qui est "bloqué" sur les lignes "AntiEntropy".

```
$ nodetool tpstats
Pool Name                    Active   Pending      Completed   Blocked  All time blocked
...
AntiEntropyStage                  0         0         854866         0                 0
...
AntiEntropySessions               0         0           2576         0                 0
...
```

Examples

Exemples d'exécution de la réparation Nodetool

Usage:

```
$ nodetool repair [-h | -p | -pw | -u] <flags> [ -- keyspace_name [table_name]]
```

Option de réparation par défaut

```
$ nodetool repair
```

Cette commande répare la plage de jetons principale du nœud actuel (c'est-à-dire la plage dont elle est propriétaire) ainsi que les répliques des autres plages de jetons qu'elle possède dans toutes les tables et tous les espaces clés du nœud actuel:

Par exemple, si vous avez un facteur de réplication de 3, alors le total de 5 nœuds sera impliqué dans la réparation: 2 nœuds répareront 1 plage de partition 2 nœuds répareront 2 plages de partition 1 nœud fixera 3 plages de partition. (La commande a été exécutée sur ce noeud)

Réparation en parallèle

```
$ nodetool repair -par
```

Cette commande s'exécutera pour effectuer la même tâche que la réparation par défaut, mais en exécutant la réparation en parallèle sur les noeuds contenant des répliques.

Réparation de la plage de jetons principale

Cette commande répare uniquement la plage de jetons principale du nœud dans toutes les tables et tous les espaces de touches sur le nœud actuel:

```
$ nodetool repair -pr
```

Réparez uniquement le centre de données local sur lequel réside le noeud:

```
$ nodetool repair -pr -local
```

Réparez uniquement la plage principale pour toutes les répliques dans toutes les tables et tous les espaces de touches sur le nœud actuel, uniquement en streaming à partir des nœuds répertoriés:

```
$ nodetool repair -pr -hosts 192.168.0.2, 192.168.0.3, 192.168.0.4
```

Réparez uniquement la plage principale de toutes les répliques dans l'espace de clés stackoverflow du nœud actuel:

```
$ nodetool repair -pr -- stackoverflow
```

Réparez uniquement la plage principale de tous les réplicas de la table test_users de l'espace de clés stackoverflow du nœud actuel:

```
$ nodetool repair -pr -- stackoverflow test_users
```

Lire Réparations à Cassandra en ligne: https://riptutorial.com/fr/cassandra/topic/4873/reparations-a-cassandra

Chapitre 7: Réparer en cours d'exécution sur Cassandra

Syntaxe

- **Synopsis**
- `nodetool [node-options] repair [other-options]`
- **Options de noeud**
- `[(-h <host> | --host <host>)]`
- `[(-p <port> | --port <port>)]`
- `[(-pw <password> | --password <password>)]`
- `[(-pwf <passwordFilePath> | --password-file <passwordFilePath>)]`
- `[(-u <username> | --username <username>)]`
- **Autres options**
- `[(-dc <specific_dc> | --in-dc <specific_dc>)...]`
- `[(-dcpar | --dc-parallel)]`
- `[(-et <end_token> | --end-token <end_token>)]`
- `[(-full | --full)]`
- `[(-hosts <specific_host> | --in-hosts <specific_host>)...]`
- `[(-j <job_threads> | --job-threads <job_threads>)]`
- `[(-local | --in-local-dc)]`
- `[(-pl | --pull)]`
- `[(-pr | --partitioner-range)]`
- `[(-seq | --sequential)]`
- `[(-st <start_token> | --start-token <start_token>)]`
- `[(-tr | --trace)]`
- `[--]`
- `[<keyspace> <tables>...]`

Paramètres

Paramètre	Détails
`-dc <specific_dc>` , `--in-dc <specific_dc>`	Utiliser `-dc` pour réparer des centres de données spécifiques
`-dcpar` , `--dc-parallel`	Utilisez `-dcpar` pour réparer des centres de données en parallèle.
`-et <end_token>` , `--end-token <end_token>`	Utilisez `-et` pour spécifier un jeton à la fin de la plage de réparation
`-full` , `--full`	Utilisez `-full` pour émettre une réparation complète.
`-h <host>` , `--host <host>`	Nom d'hôte ou adresse IP du noeud
`-hosts <specific_host>` , `--in-hosts <specific_host>`	Utilisez `-hosts` pour réparer des hôtes spécifiques
`-j <job_threads>` , `--job-threads <job_threads>`	Nombre de threads pour exécuter les travaux de réparation. Habituellement, cela signifie le nombre de CF à réparer simultanément. AVERTISSEMENT: augmenter cela met plus de charge sur la réparation des nœuds, alors soyez prudent. (par défaut: 1, max: 4)
`-local` , `--in-local-dc`	Utilisez `-local` pour réparer uniquement les nœuds du même centre de données
`-p <port>` , `--port <port>`	Numéro de port de l'agent jmx distant
`-pl` , `--pull`	Utilisez l' `--pull` pour effectuer une réparation unidirectionnelle où les données ne sont transmises que d'un noeud distant à ce noeud.
`-pr` , `--partitioner-range`	Utilisez `-pr` pour réparer uniquement la première plage renvoyée par le partitionneur
`-pw <password>` , `--password <password>`	Mot de passe de l'agent jmx distant
`-pwf <passwordFilePath>` , `--password-file <passwordFilePath>`	Chemin d'accès au fichier de mot de passe JMX
`-seq` , `--sequential`	Utilisez `-seq` pour effectuer une réparation séquentielle
`-st <start_token>` , `--start-token <start_token>`	Utilisez `-st` pour spécifier un jeton au début de la plage de réparation
`-tr` , `--trace`	Utilisez `-tr` pour suivre la réparation. Les traces sont enregistrées dans `system_traces.events` .
`-u <username>` , `--username <username>`	Nom d'utilisateur de l'agent jmx distant

Paramètre	Détails
`--`	Cette option peut être utilisée pour séparer les options de ligne de commande de la liste des arguments (utile lorsque des arguments peuvent être pris pour des options de ligne de commande)
`[<keyspace> <tables>...]`	L'espace de clés suivi d'une ou plusieurs tables

Examples

En cours de réparation sur Cassandra

Exécutez la réparation sur une plage de partition particulière.

```
nodetool repair -pr
```

Exécutez la réparation sur l'ensemble du cluster.

```
nodetool repair
```

Exécutez la réparation en mode parallèle.

```
nodetool repair -par
```

Lire Réparer en cours d'exécution sur Cassandra en ligne:
https://riptutorial.com/fr/cassandra/topic/6556/reparer-en-cours-d-execution-sur-cassandra

Chapitre 8: Sécurité

Remarques

Ressources de sécurité Cassandra

- CQL: définition de la syntaxe des rôles de base de données
- CQL: Liste des autorisations d'objet
- Documentation DataStax: Authentification interne
- Documentation DataStax: autorisation interne

Examples

Configuration de l'authentification interne

Cassandra n'exige pas que les utilisateurs se connectent en utilisant la configuration par défaut. Au lieu de cela, les connexions anonymes sont autorisées pour toute personne pouvant se connecter au port `native_transport_port` . Ce comportement peut être modifié en éditant la configuration de `cassandra.yaml` pour utiliser un authentificateur différent:

```
# Allow anonymous logins without authentication
# authenticator: AllowAllAuthenticator

# Use username/password based logins
authenticator: PasswordAuthenticator
```

Les informations d'identification de connexion validées par `PasswordAuthenticator` seront stockées dans l' `system_auth` clés interne system_auth. Par défaut, l'espace de clés ne sera pas répliqué sur tous les nœuds. Vous devrez modifier les paramètres de réplication pour vous assurer que Cassandra sera toujours en mesure de lire les informations d'identification de l'utilisateur à partir du stockage local au cas où d'autres nœuds du cluster ne pourraient pas être atteints.

Pour `SimpleStrategy` (où `N` est le nombre de nœuds de votre cluster):

```
ALTER KEYSPACE system_auth WITH replication = {'class': 'SimpleStrategy',
'replication_factor': N};
```

Pour `NetworkTopologyStrategy` (où `N` est le nombre de nœuds du centre de données correspondant):

```
ALTER KEYSPACE system_auth WITH replication =  { 'class' : 'NetworkTopologyStrategy',
'datacenter1' : N };
```

Redémarrez chaque nœud après les modifications décrites ci-dessus. Vous ne pourrez plus vous connecter qu'avec le superutilisateur par défaut:

```
cqlsh -u cassandra -p cassandra
```

(Facultatif) Remplacez le superutilisateur par défaut par un utilisateur personnalisé

Utiliser un super-utilisateur par défaut avec un mot de passe standard n'est pas plus sûr que d'utiliser aucun utilisateur. Vous devez créer votre propre utilisateur en utilisant un mot de passe sûr et unique:

```
CREATE ROLE myadminuser WITH PASSWORD = 'admin123' AND LOGIN = true AND SUPERUSER = true;
```

Connectez-vous avec votre nouvel utilisateur: `cqlsh -u myadminuser -p admin123`

Désactivez maintenant la connexion pour l'utilisateur standard de cassandra et supprimez le statut de superutilisateur:

```
ALTER ROLE cassandra WITH LOGIN = false AND SUPERUSER = false;
```

Configuration de l'autorisation interne

Par défaut, chaque utilisateur pourra accéder à toutes les données de Cassandra. Vous devrez configurer un autorisateur différent dans votre `cassandra.yaml` pour accorder des autorisations d'objet individuelles à vos utilisateurs.

```
# Grant all permissions to all users
# authorizer: AllowAllAuthorizer

# Use object permissions managed internally by Cassandra
authorizer: CassandraAuthorizer
```

Les autorisations pour des utilisateurs individuels seront `system_auth` dans l' `system_auth` clés interne system_auth. Vous devez modifier les paramètres de réplication au cas où vous ne l'auriez pas déjà fait en activant l'authentification par mot de passe.

Pour `SimpleStrategy` (où `N` est le nombre de nœuds de votre cluster):

```
ALTER KEYSPACE system_auth WITH replication = {'class': 'SimpleStrategy',
'replication_factor': N};
```

Pour `NetworkTopologyStrategy` (où `N` est le nombre de nœuds du centre de données correspondant):

```
ALTER KEYSPACE system_auth WITH replication =  { 'class' : 'NetworkTopologyStrategy',
'datacenter1' : N };
```

Redémarrez chaque nœud après les modifications décrites ci-dessus. Vous pourrez désormais définir des autorisations à l'aide, par exemple, des commandes suivantes.

Accorde toutes les autorisations pour l'espace de clés et le rôle spécifiés:

```
GRANT ALL ON KEYSPACE keyspace_name TO role_name;
```

Accordez des autorisations de lecture sur tous les espaces clés:

```
GRANT SELECT ON ALL KEYSPACES TO role_name;
```

Autoriser l'exécution des instructions INSERT, UPDATE, DELETE et TRUNCATE sur un certain espace de clés:

```
GRANT MODIFY ON KEYSPACE keyspace_name TO role_name;
```

Autoriser la modification des espaces clés, des tables et des index pour certains espaces clés:

```
GRANT ALTER ON KEYSPACE keyspace_name TO role_name;
```

S'il vous plaît noter que les autorisations seront mises en cache pour `permissions_validity_in_ms` (`cassandra.yaml`) et les modifications peuvent ne pas être efficaces instantanément.

Lire Sécurité en ligne: https://riptutorial.com/fr/cassandra/topic/5646/securite

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec Cassandra	Community, muru, perennial_noob, phact, Ravinder Matte, sourav, Stephen Leppik
2	Cassandra - PHP	Aaron, ethrbunny, Renjith VR
3	Cassandra en tant que service	Shoban Sundar
4	Clés Cassandra	Derlin
5	Connexion à Cassandra	geeves
6	Réparations à Cassandra	Aaron, Akshay
7	Réparer en cours d'exécution sur Cassandra	muru, Ravinder Matte
8	Sécurité	Stefan Podkowinski