Formation Pratique Cassandra

Ynov 2025

3 février 2025

L'objectif de ce TP est une prise en main du SGBD Cassandra ; nous allons aborder les points suivants :

1. Interaction avec le moteur C* en ligne de commande
2. Création , peuplement et interrogation de la base de données
3. l'outil natif de supervision Nodetool et Administration
4. Migration des données
5. Rotation des logs des logs
6. Configuration JVM

---

1 Installation et configuration de Cassandra

La version 5 est sortie au mois de septembre et comporte de nombreuses évolutions. Nous pouvons l'utiliser au cours de cette formation.

Pour ceux qui le souhaitent, il est aussi possible d'utiliser la version 4.

Dans un premier temps, nous allons nous assurer que les outils nécessaires, pour le fonctionnement de C*, sont déjà installés.

1.0.1 Installation des pré-requis

Installation de java

Java 8 est toujours possible dans la version 4. Toutefois, l'utilisation de java11 est recommandée.

1. Vérifier la version de votre version java.

```
Java -version
openjdk version "11.0.8" 2020-07-14
OpenJDK Runtime Environment AdoptOpenJDK (build 11.0.8+10)
OpenJDK 64-Bit Server VM AdoptOpenJDK (build 11.0.8+10, mixed mode)
```

2. En cas de plusieurs versions, vous pouvez spécifier celle à utiliser :

```
export JAVA_HOME=‘/usr/libexec/java_home -v 1.8‘
```

Installation de python

1. les instructions à suivre

```
cd /usr/src
wget https://www.python.org/ftp/python/2.7.12/ Python-2.7.12.tgz
tar xzf Python-2.7.12.tgz
cd Python-2.7.12
sudo ./configure
sudo make altinstall
python2.7 -V Python 2.7.13
```

1.0.2 Installation Cassandra

Vous pouvez trouver le binaire à l'adresse suivante :
https ://www.apache.org/dyn/closer.lua/cassandra/4.0-beta2/apache-cassandra-4.0-beta2-bin.tar.gz

Noter qu'il est possible de passer par une installation en utilisant les packages.

```
 -- Dans le fichier
    nano /etc/yum.repos.d/cassandra.repo

-- Ajouter
[cassandra]
name=Apache Cassandra
baseurl=https://www.apache.org/dist/cassandra/redhat/311x/
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://www.apache.org/dist/cassandra/KEYS

-- Recharger les deamons
yum -y install cassandra

-- Lancer le service de cassandra
systemctl daemon-reload

\\
```

1.0.3 RÉPERTOIRES D'INSTALLATION

Dans un premier temps, nous devons créer les répertoires nécessaires. Par défaut, ces répertoires sont :

- /var/lib/cassandra/commitlog
- /var/lib/cassandra/data
- /var/log/cassandra/

Pensez donc à les créer et à leur affecter les droits nécessaires (on suppose que l'on utilise le user 'cassandra') :

```
    sudo mkdir \/{/var/lib/cassandra} \\
    sudo mkdir \/{/var/lib/cassandra/commitlog} \\
    sudo mkdir \/{/var/lib/cassandra/data } \\
    sudo mkdir \/{/var/log/cassandra/} \\
    sudo chown -R cassandra:cassandra \/{ /var/lib/cassandra/ /var/log/cassandra/ }
```

Cassandra propose un fichier cassandra.yaml, situé dans le dossier $CASSANDRA_HOME/ (Répertoire conf). Lorsque vous commencez avec Cassandra, vous pouvez simplement modifier quelques propriétés de configuration pour démarrer. Plus tard, une fois que vous aurez bien compris l'architecture de Cassandra et les concepts clés, il sera important de se focaliser davantage sur les propriétés de configuration qui affectent les performances et de nombreux autres aspects.

Nous allons configurer les paramètres suivants :
- cluster_name
- listen_adresss
- listen_interface

1.0.4 Configuration du Firewall

Avant de démarrer l'instance Cassandra, assurez-vous que vous pouvez accéder aux services Cassandra depuis l'extérieur, en ouvrant ces ports :
— 7000
— 7199
— 9042
— 9160

Dans le cas d'une utilisation locale, cette vérification devient secondaire.

1.0.5 Démarrage de cassandra

Pour démarrer Cassandra, lancer le daemon C* depuis le répertoire $CASSANDRA_HOME/bin/

```
[cqlsh 5.0.1 | Cassandra 4.0-beta2 | CQL spec 3.4.5 | Native protocol v4]
Use HELP for help.
cqlsh>
```

Vérifiez l'état de votre cluster cassandra :

```
mohammed@MacBook-Pro-de-Mohammed ~ % nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens       Owns (effective)  Host ID
Rack
UN  127.0.0.1  667,84 KiB  256          100,0 \%      fea84f37-a99b-4e62-ba23-4b rack1
```

---

2 Se familiariser avec l'invite CQLSH

2.1 Prise en main

1. Vérifier le niveau de consistance (CL - consistency level)
2. Mettre le CL à Local-quorum (puis à ONE)
3. Vérifier à quel Data-Center vous êtes connectés
4. Consulter les différents keyspaces déja présents

5. Consulter l'heure interne de la base cassandra
6. Consulter l'heure de votre machine locale et la comparer avec celle de cassandra
7. Quel est le nom de votre host ?
8. Modifier l'output pour le mettre en 'output vertical'
9. . Donner la version de cqlsh que vous êtes entrain d'utiliser

2.2 Configuration de son environnement de variable local

Dans cette partie nous allons nous interesser au fichier cqlshrc qui se trouve dans le dossier ' /.cassandra/cqlshrc'. Parfois ce dernier n'existe pas, il sera donc nécessaire de le créer.

Un point de rappel est nécessaire avant d'entamer cette partie. Nous allons nous intéresser aux sections de base, à savoir : [csv] [connection] [ui]

1. Fixer le format de de la date
2. Fixer la timezone en 'Europe/Paris'
3. Vérifier à quel Data-Center vous êtes connectés
4. Consulter les différents keyspaces déja présents
5. Enregistrer vos identifiants pour vous affranchir de les sasiir à chaque connexion
6. enregistrer votre host de connexion par défaut

3 Interrogation de Cassandra

3.1 Création de la base de données

3.1.1 keyspace

Nous allons dans un premier temps créer un keyspace (avec RF = 3 si cela est possible) nommé 'Ecole' mais que vous pouvez, pour le bon déroulement du TP, le nommer autrement, par exemple 'Ecole_votre_nom'.

Selon l'architecture du cluster, la stratégie de réplication peut différer. penser à donc utiliser la bonne stratégie.

=> Quelle est la stratégie que vous avez utilisée et pourquoi ?

1. lister les keyspaces présents.
2. orienter la lister des ks dans un fichier txt.
3. Créer votre keyspace avec un RF = 3.
4. Connectez-vous au keyspace (commande use).
5. Regarder les infos relatives à votre cluster (Desc).
6. consulter la topologie

7. vérifier la valeur de votre RF

```
CREATE  KEYSPACE [IF NOT EXISTS] keyspace_name
   WITH REPLICATION = {
      'class' : 'SimpleStrategy', 'replication_factor' : N }
     | 'class' : 'NetworkTopologyStrategy',
       'dc1_name' : N [, ...]
   }
   [AND DURABLE_WRITES =  true|false] ;
```

Pour rappel, pour lister l'ensemble des schémas des keyspaces, vous pouvez utiliser la commande :

```
SELECT * FROM system.schema_keyspaces;
```

Ainsi l'ensemble des informations sont hébergées dans le keyspace *System*.

3.2 Création des tables

Cql3 est le langage d'interrogation natif de Cassandra, il utilise la même syntaxe que SQL mais avec des différences fondamentales que nous verrons dans la suite de ce TP.

Nous créons deux tables (ou Column Family sous Cassandra) *Cours* et *Enseignant* avec les structures suivantes :

```
CREATE TABLE Cours (
idCours INT,
Intitule VARCHAR,
Responsable INT,
Niveau VARCHAR,
nbHeuresMax INT,
Coeff INT,
PRIMARY KEY ( idCours )
);
CREATE TABLE Enseignant (
idEnseignant INT PRIMARY KEY,
Nom VARCHAR,
Prenom VARCHAR,
status VARCHAR
 );
```

1. Afficher la structure des tables.
2. Regarder l ordre des champs (correspond-il à l'ordre de la structure ?).
3. Donner le nombre de tables se trouvant dans le ks ecole
4. Quelle est la durée de vie des tombstones relatives à chacune de vos tables

3.2.1 Insertion des données

Nous allons dans cette partie procéder au peuplement de nos tables.
Comme en SQL, il existe différentes options d'insertion des données :

1. Directement dans le terminal CQLSH

   ```
   INSERT INTO nomtable (nomscolonnes...) VALUES (valeurs...);
   ```

2. En appelant un script CQL en utilisant la fonction '*Source*' suivie du 'chemin_du_fichier'.

```
##### table cours
INSERT INTO cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (
1,'Introduction␣aux␣Bases␣de␣Donnees',1,'M1',30,3);
INSERT INTO cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (
2,'Immeubles␣de␣Grandes␣Hauteurs',4,'M1',30,2);
INSERT INTO cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (
3,'Production␣et␣distribution␣de␣biens␣et␣de␣ser',5,'M1',30,2);
INSERT INTO cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (
4,'Bases␣de␣Donnees␣Avancees',1,'M2',30,5);
INSERT INTO cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (
5,'Architecture␣des␣Systemes␣Materiel',6,'M2',8,1);
INSERT INTO cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (
6,'IT␣Business␣/␣Introduction',7,'M2',20,3);
INSERT INTO cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (
7,'IT␣Business␣/␣Strategie␣et␣Management',8,'M2',10,1);
##### table Enseignant
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (
1,'Travers','Nicolas','Vacataire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (
2,'Mourier','Pascale','Titulaire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (
3,'Boisson','Francois','Vacataire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (
4,'Mathieu','Eric','Titulaire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (
5,'Chu','Chengbin','Titulaire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (
6,'Boutin','Philippe','Titulaire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (
7,'Escribe','Julien','Vacataire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (
8,'Znaty','David','Vacataire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (
9,'Abal-Kassim','Cheik␣Ahamed','Vacataire');
```

3. En utilisant la fonction *Copy* :

```
COPY table_name [( column_list )]
FROM 'file_name'[, 'file2_name', ...] | STDIN
[WITH option = 'value' [AND ...]]
```

les données du script csv à construire.

idCours ,Intitule ,Responsable ,Niveau ,nbHeuresMax ,Coeff
1,'Introduction␣aux␣Bases␣de␣Donnees ',1,'M1' ,30,3
2,'Immeubles␣de␣Grandes␣Hauteurs ',4,'M1' ,30,2
3,'Production␣et␣distribution␣de␣biens␣et␣de␣ser',5,'M1' ,30,2
4,'Bases␣de␣Donnees␣Avancees ',1,'M2' ,30,5
5,'Architecture␣des␣Systemes␣Materiel ',6,'M2' ,8,1
6,'IT␣Business␣/␣Introduction ',7,'M2' ,20,3
7,'IT␣Business␣/␣Strategie␣et␣Management ',8,'M2' ,10,1

3.2.2 Interrogation

Avant de manipuler les données, nous nous arrêtons un peu sur les estimations de nos tables : le nombre de partitions et la taille moyenne des partitions par table.

— au niveau de la table size estimates du keyspace system

— la taille moyenne des partitions par table.
— quelle information pertinente retenir de cette table ?

Pour interroger la base de données, la syntaxe CQL est fortement inspirée de SQL. Pour la suite des exercises, exprimer en CQL la requête demandée :

1. Lister les intitulés des cours ;
2. Donner le nom de l'enseignant dont l'identifiant est 4 ;

3.3 cql avancé

Dans cette section nous allons découvrir les index et les types complexes.

3.3.1 GESTION DES INDEX

Si vous utilisez la version 5, il est préfeable d'utiliser la type d'index SAI. sinon, on peut utiliser les index SASI à activer au préalble dans le fichier de configuration (SASI = TRUE).

Pour rappel, SASI est un type d'index secondaire développé par Apple et mis à disposition de la communauté Cassandra (introduit depuis Cassandra 3.4 mais je recommande Cassandra 3.5 au moins à cause de corrections de bugs critiques). il existe trois modes :

1. PREFIX - Utilisé pour servir les requêtes LIKE en fonction du préfixe de la colonne indexée
2. CONTAINS - Utilisé pour répondre aux requêtes LIKE selon que le terme recherché existe ou non dans la colonne indexée.
3. SPARSE - Utilisé pour indexer les données rares . Par exemple, les requêtes de plage qui couvrent des horodatages volumineux.

Pour effectuer les requêtes suivantes, vous auriez besoin de créer un index. on préférera l'index SASI.

Notez qu'il est possible d'utiliser des index secondaires classiques.

Pour rappel, les index secondaires sont à utiliser avec prudence. Discuter avec le formateur les limites de leur utilisation dans une architecture distribuée.

```
CREATE INDEX Enseignement_{idx} ON Cours ( Responsable ) ;
```

1. Intitulé des cours du responsable numéro 1 ;
2. Intitulé des cours dont le nombre d'heures maximum est égal à 30 ;
3. Intitulé des cours dont le responsable '1' et dont le niveau est 'M1' ; Utiliser 'ALLOW FILTERING'.
4. Intitulé des cours dont les responsables ont une valeur inférieure à 5 ;
5. les requêtes suivantes retournent une erreur comment y remédier ?

```
select idcours , max ( nbHeuresMax ) from cours group by idCours , Niveau ;
select * from cours  order by  Niveau ;
```

3.3.2 Manipuler des types complexes

Dans cet exercice, l'objectif est de manipuler les collections avec le moteur C*. Le choix du type est laissé à votre appréciation.

Dans le table cours, nous souhaitons ajouter une colonne pour y stocker des labels inhérents au modules enseignés. l'ordre des labels n'a aucune importance ; l'ordre de restitution non plus.

Bases de Données Avancées : triggers, procédures stockées

Introduction aux Bases de Données : SGBD, Normalisation, SQL

Immeubles de Grandes Hauteurs : Architecture, Génie civile

Nous souhaitons désormais introduire la notion d'ordre. Supprimer les éléments de la colonne Labels pour les renseigner avec cette nouvelle contrainte.

Ajouter une nouvelle colonne appelée details pour y inserer le nombre d'heures pour chaque labels :

triggers : 10H, procédures stockées : 15H

SGBD 20H, Normalisation 15H, SQL : 10H

Immeubles de Grandes Hauteurs : Architectures modernes 25H , Génie civile 25H

Bases de Données Avancées : triggers, procédures stockées

Introduction aux Bases de Données : SGBD, Normalisation, SQL

Immeubles de Grandes Hauteurs : Architecture, Génie civile

3.3.3 Apprendre à éviter les jointures

La jointure n'est pas possible avec CQL (à cause du stockage par hachage distribué). Nous allons utiliser une approche pour dénormaliser deux tables et qui repose sur l'utilisation des listes. On va intégrer la table 'Enseignant' dans la table 'Cours', nous appellerons cette table '*coursEnseignant*'.

Premièrement on va créer le type enseignant. en effet, Cassandra nous offre la possibilité de créer notre propre type de données. nous allons nous en servir pour la suite de l'exercice.

```
CREATE TYPE Enseignant (
idEnseignant , Nom VARCHAR, Prenom VARCHAR, status VARCHAR,
);
```

Créer une nouvelle table *CoursEnseignant*. Cette table va contenir en plus des attributs de la table cours, les attributs de la table enseignants. Pensez à utiliser une liste dans laquelle on stocke les éléments de type enseignant (un cours peut avoir plusieurs responsables).

```
### exemple
CREATE TABLE mykeyspace.users (
  id uuid PRIMARY KEY,
  name frozen <fullname>,
  direct_reports set<frozen <fullname>>,     // a collection set
  addresses map<text, frozen <address>>     // a collection map
  score set<frozen <set<int>>>              // a set with a nested frozen set
);
```

Maintenant nous allons insérer les données dans la table *CoursEnseignant* :

```
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff)
VALUES(1,'Introduction␣aux␣Bases␣de␣Donnees', {'idenseignant':'1','nom':'Travers'
,'prenom':'Nicolas','status':'Vacataire'},'M1',30,3);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,
Coeff) VALUES(2,'Immeubles␣de␣Grandes␣Hauteurs',
{'idenseignant':'4','nom':'Mathieu','prenom':'Eric','status':'Titulaire'},
'M1',30,2);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,
nbHeuresMax,Coeff) VALUES(3,'Production␣et␣distribution␣de␣biens␣et␣de␣ser',
{'idenseignant':'5','nom':'Chu','prenom':'Chengbin','status':'Titulaire'},'M1'
,30,2);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,
Coeff) VALUES(4,'Bases␣de␣Donnees␣Avancees',
{'idenseignant':'1','nom':'Travers','prenom':'Nicolas','status':'Vacataire'},
'M2',30,5);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,
Coeff) VALUES(5,'Architecture␣des␣Systemes␣Materiel',
{'idenseignant':'6','nom':'Boutin','prenom':'Philippe','status':'Titulaire'},
'M2',8,1);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,
Coeff) VALUES(6,'IT␣Business␣/␣Introduction',
{'idenseignant':'7','nom':'Escribe','prenom':'Julien','status':'Vacataire'},
'M2',20,3);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax
,Coeff) VALUES(7,'IT␣Business␣/␣Strategie␣et␣Management',
{'idenseignant':'8','nom':'Znaty','prenom':'David','status':'Vacataire'},
'M2',10,1);
```

1. Créer un index sur le champ Niveau de la table CoursEnseignant ;

Parfois, pour éviter de recréer la table (pour contourner les limites CQL), on peut utiliser les vues. On va créer une vue matérialisée de la table cours en ajoutant le champs *niveau* dans la clé de partitionnement.

```
CREATE MATERIALIZED VIEW ecole.cours_niveau
AS SELECT * FROM ecole.cours
WHERE idcours IS NOT NULL AND niveau IS NOT NULL
PRIMARY KEY (idcours, niveau)
```

Pour afficher toutes les vues on utilise la commande suivante :

```
select * from system.built_view
```

Maintenant on lance la même requête de grpupe by mais cette fois-ci sur la vue coursniveau. est possible.

```
select idcours,sum(nbHeuresMax) from cours_niveau group by idCours,Niveau;
```

Utilisation d'une collection pour archiver les données des tables cours et enseignants. le type list serait pertinent.

L'idée ici est de créer une table qui permet de tracer les données des données tables selon la structure ci-dessous. Chaque ligne de la table legacy représente un élément de la collection.

```
CREATE TABLE archive_cours ( tabname text  PRIMARY KEY,
datemaj timestamp,
params list<text>,
LAST_UPDATE timestamp);
```

Objectif Veuillez exporter via la commande copy les données des deux tables en csv. En utilisant un script shell, veuillez peupler cette nouvelle table depuis les fichiers csv.

l'outil dsbulk est efficace pour ce genre d'opérations.

4 Administration

Dans cette section, nous verrons les trois aspects suivants :

Sauvegarde des données

Import Export

Restauration des données

utilisation de cassandra-reaper

configuration d'un environnemnt de développent

Gestion de la timezone

4.1 Slow queries

Modifier la configuration cassandra.yaml pour collecter les requêtes longues.
slow_query_log_timeout_in_ms : 200

Exécuter quelques requêtes avec un count et regarder ce qui se passe dans le fichier debug.log

4.2 Sauvegarde et restauration

4.2.1 Sauvegarde

L'objectif ici est d'automatiser la prise de snapshots.

Proposer un script python (shell, ou autre) pour gérer automatiquement la sauvegarde des données et la suppression des snapshots selon la stratégie suivante :

Chaque dimanche à 23H, une sauvegarde des données est effectuée. Les snapshots doivent être nommés selon une règle de votre choix. L'utilisation de la date du jour est un bon choix.

Penser à nodetool snapshot –tag <tag> –<keyspace>

l'automatisation doit se faire via l'utilisation d'un cron.
https ://www.digitalocean.com/community/tutorials/how-to-use-cron-to-automate-tasks-centos-8-fr

4.2.2 Restauration

Dans cet exercice nous allons simuler la perte d'une machine et nous tenterons de la restaurer depuis une sauvegarde locale.

1. Faire une sauvegarde des données du keyspace courant (celui que vous utilisez)
2. Faire une sauvegarde du modèle de données (desc keyspace keyspace > MDD.csv)
3. Arrêter Cassandra
4. Vider le répertoire des logs
5. Renommer le répertoire path-cassandra/data de stockage en data-bcp
6. Créer un répertoire data
7. Redémarrer cassandra
8. Consulter vos données. Quel constat ?
9. Créer le modèle de données
10. Restaurer le keyspace courant via sstableloader

4.3 Import Export

Le but de cette partie a un objectif double :

1. Explorer les outils possibles
2. déterminer lequel choisir et dans quels cas ?

Rappelons que les outils communement utilisés sont :

- Commande native copy
- Outil Dsbulk
- SStableLoader
- Depuis des sstables produits via cqlwriter
- Spark

Commande copy Charger les données selon les recommandations suivantes :

- Chargement le fichier calendar avec paramétrage par défaut
- Chargement en utilisant des paramètres optimisés
- Comparer le temps de chargement des des opérations

Voici la structure de la table à créer pour pouvoir charger les données

```
    CREATE TABLE calendar (
race_id int,
race_name text,
race_start_date timestamp,
race_end_date timestamp,
PRIMARY KEY (race_id, race_start_date, race_end_date))
WITH CLUSTERING ORDER BY (race_start_date DESC, race_end_date DESC);
```

avec DSbulk

En utilisant l'outil dsbulk : - Exporter en format csv les données des deux tables cours et enseignants
- Charger la table calendar et puis compter le nombre de lignes (que constatez-vous ?)
- Charger calendar avec copy et comparer les temps
- Exporter la table calendar et puis importer la dans un autre keyspace que vous auriez créé au préalable. L'opération doit être faite avec une seule commande. Donner également le nombre de lignes insérées.

4.4 Environnemt de test

L'idée ici est de mener une réflexion sur les paramétres que l'on peut se permettre de modifier dans un environnement de développement.

Plusieurs niveaux peuvent être regardés :
- Gestion de la mémoire - Gestion des bloom filters et donc des tombsones - Gestion du stockage

4.5 La maintenance avec cassandra reaper

Vous allons apprendre à utiliser la planification des opération de maintenance en utilisant l'outil Cassandra-reaper. //

1. Télecharger et installer l'outil
2. Lancer le daemon reaper depuis le dossier des binaires
3. Connecter-vous depuis la page localhost :8080/webui/
4. Ajouter votre cluster cassandra
5. modifier le numéro de port dans le fichier de configuration
6. reconnecter-vous via la même page web et ajouter le cluster
7. Lancer un reaper sur votre cluster
8. lancer une sauvegarde et vérifier si cette dernière a été reussie.
9. planifier un rapair chaque jour.

4.5.1 État du cluster et monitoring

Cassandra offre deux outils fondamentaux le premier est *cqlsh* celui utilisé dans la première parties de ce tp il nous permet de la gestion de base de donnée ainsi de définir les règles de Consistency. le second outil c'est *nodetool*.
L'utilitaire nodetool est un excellent outil d'administration et de surveillance. La section suivante abordera certaines des fonctionnalités utiles de nodetool. La plupart des commandes nodetool sont évidentes et peuvent être facilement apprises en lisant le texte d'aide. on va voir dans cette section quelques informations relatives à l'état du cluster.

Il y a d'autres commandes, mais destinées à l'administrateur seulement. Ne tentez pas de les lancer, car si vous cassez le cluster, plus personne ne pourra travailler.

```
nodetool status
nodetool info
nodetool netstats
nodetool tablestats
```

1. Vérifier l'état de chaque noeud.
2. quel est la charge de données hébergé par chaque noeud.
3. quel est protocole de communication utilisé ?
4. vérifier les activités de compactions
5. Vérifier si des écritures (mutations) ont été refusées).

5 Modélisation des données dans C*

5.0.1 Initiation

Nous allons dans un premier temps nous arrêter sur les limites du langage CQL pour bien comprendre le principe de 'Query-Driven Data Modeling'

Nous allons voir plusieurs modélisation relatives à une étude de températures observées par ville.

5.0.2 Initiation

Voici le modèle de données de cette première modélisation,

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

CREATE TABLE temperature1 (ville text,temperature int, PRIMARY KEY (ville));

INSERT INTO temperature1(ville, temperature ) VALUES ( 'Paris', 30);
INSERT INTO temperature1(ville, temperature ) VALUES ( 'Paris', 29);
INSERT INTO temperature1(ville, temperature ) VALUES ( 'Rennes', 30);

 \\
```

Voici le besoin pour cette première modélisation,

SELECT * FROM temperature1 ; – OK
SELECT * FROM temperature1 WHERE ville='Paris' ;
SELECT * FROM temperature1 WHERE temperature=30 ;
SELECT * FROM temperature1 WHERE temperature=30 ;
SELECT * FROM temperature1 WHERE ville='Paris' AND temperature>20 ;
SELECT count(*) FROM temperature1 WHERE ville='Paris' ;
SELECT ville, max(temperature) FROM temperature1 GROUP BY ville ;

Question : Quelle est la limite de ce modèle ?

Voici le modèle de données de la seconde modélisation. On souhaite désormais surveiller les températures par ville et par date.

```
 CREATE TABLE temperature2 (ville text,record_date text,temperature int,
 PRIMARY KEY (ville, record_date))
 WITH CLUSTERING ORDER BY (record_date DESC) ;

INSERT INTO temperature2 (ville, record_date, temperature )
VALUES ( 'Paris', '2017/11/14', 30);

INSERT INTO temperature2 (ville, record_date, temperature )
VALUES ( 'Paris', '2017/11/13', 29);

INSERT INTO temperature2 (ville, record_date, temperature )
VALUES ( 'Rennes', '2016/11/10', 30);

INSERT INTO temperature2 (ville, record_date, temperature )
VALUES ( 'Paris', '2017/11/15', 29);


 \\
```

Question : Quelle est la limite de ce modèle ?

Voici le modèle de données de la seconde modélisation. On souhaite désormais surveiller les températures par ville et par date.

```
 CREATE TABLE temperature3 (ville text,
        jour text,
        heure text,
        temperature int,
        PRIMARY KEY ((ville, jour), heure)
);

INSERT INTO temperature3 (ville, jour, heure, temperature)
VALUES ('Paris', '2014-01-20', '8', 7);
INSERT INTO temperature3 (ville, jour, heure, temperature)
VALUES ('Paris', '2014-01-20', '9', 8);
INSERT INTO temperature3 (ville, jour, heure, temperature)
VALUES ('Paris', '2014-01-20', '10', 7);
INSERT INTO temperature3 (ville, jour, heure, temperature)
VALUES ('Paris', '2014-01-21', '10', 7);

\\
```

Veuillez exécuter les requêtes suivantes. Essayer de comparer la taille des partitions des trois modélisations.

5.0.3 Modélisation C* Dans un projet à Grande échelle

Dans cette section, nous allons nous confronter aux contraintes de modélisation que l'on rencontre dans un projet avec un architecture opérationnelle.

On se propose deux cas de figures :

cas 1 Proposer une modélisation orientée C* en vous basant sur les requêtes métiers.

Dans cet exercice on suppose que l'on est dans cadre de modernisation stricte où nous devons garder la même cardinalité : une table legacy correspond à une table Cassandra. Cas 1 : proposer une modélisation orientée C* en vous basant sur les requêtes métiers.

Dans cette exercice on suppose que l'on est dans cadre de modernisation stricte où nous devons garder la même cardinalité : une table legacy correspond à une table Cassandra.

cas 2 Proposer une modélisation orientée Cassandra en vous basant sur les requêtes métiers

Dans cette exercice on suppose que l'on est dans cadre de modernisation libre où nous pouvons revoir intégralement le modèle. Autrement dit, il est possible de dénormaliser et imaginer des solutions optimales pour supporter les requêtes,

Voici les besoins devant être supportés par votre modélisation.

Noms des logiciels UNIX

Type du poste p8

Nom, adresse IP, numéro de salle des postes de type UNIX ou PCWS

les postes du segment 130.120.80 triés par numéro de salle décroissant

Numéros des logiciels installés sur le poste p6

Numéros des postes qui hébergent le logiciel log1.

Nom et adresse IP complète (ex : 130.120.80.01) des postes de type TX

Type du poste p8

Le nombre de logiciels par poste

Noms des salles ou l'on peut trouver au moins un poste hébergeant 'Oracle 6'

L'ensemble des requêtes relatives à cette étude sont :

```
--1     Type du poste p8
SELECT nPoste, typePoste
  FROM Poste WHERE nPoste = 'p8';



--3     Nom, adresse IP, num ro de salle des postes de type UNIX ou PCWS.
SELECT nomPoste, indIP, ad, nSalle
  FROM poste
  WHERE typePoste = 'UNIX'
  OR    typePoste = 'PCWS';

--4     M me requ te pour les postes du segment 130.120.80 tri s
--par num ro de salle d croissant
SELECT nomPoste, indIP, ad, nSalle
  FROM poste
  WHERE (typePoste = 'UNIX'
  OR    typePoste = 'PCWS')
  AND   indIP = '130.120.80'
        ORDER BY nSalle DESC;


---------
  SELECT nPoste FROM Installer
  GROUP BY nPoste
  HAVING COUNT(nLog)=2;

-----------------------------------------------


  --2 Noms des logiciels UNIX
SELECT nomLog
  FROM Logiciel
```

```
  WHERE typeLog = 'UNIX';
  --11
SELECT AVG(prix)
  FROM Logiciel
  WHERE typeLog = 'UNIX';

--12
SELECT MAX(dateAch)
  FROM Logiciel;



--installer
  --10 elle ne passera jamais
SELECT nLog, COUNT(nPoste)
  FROM Installer
  GROUP BY (nLog);


--Le nombre de logiciels par poste
-8
SELECT nPoste, COUNT(nLog)
  FROM installer
  GROUP BY (nPoste);


  --20
SELECT  nomSegment
  FROM  Segment
  WHERE indIP IN
  (SELECT   indIP
   FROM     Poste
   WHERE    typePoste = 'TX'


--31 Noms des salles ou  l on  peut trouver au moins un poste h bergeant   Oracle   6
SELECT  nomSalle
  FROM  Salle NATURAL JOIN  Poste
  NATURAL JOIN  Installer
    NATURAL JOIN  Logiciel
     WHERE nomLog  = 'Oracle 6';
```

6 Configuration de de la JVM (Garbage Collector)

Par défaut, Cassandra utilise le Garbage Collector CMS

— Vérifiez la mémoire min et max allouée par défaut lors de l'utilisation de CMS
— Dans quel cas le CMS est recommandé ?
— Désactiver le CMS et activer le G1 ?
— Quelles sont les différences entre CMS et G1 ?
— Que pouvons-nous modifier, de plus, pour une gestion contrôlée des ressources mémoires ?

Remarque : toute modification du GC nécessite un stop/Start du serveur C*.

6.1 Rotation des Logs

La gestion des fichiers de log dans cassandra repose sur l'utilisation de log4j. il est possible de gérer les fichiers par taille, par dure d'expiration... mais aussi par niveau. Nous allons procéder à la configuration des fichiers Debug.log et System.log.

— Mettez les logs au niveau de votre choix (INFO, DEBUG,...) ; Gestion à faire au niveau du fichier de configuration et non en ligne de commande.

— Configurer la rotation des logs :

  — Les fihciers doivent suivre une rotation selon une taille données (en MB, GB,...)

  — Définition du nombre de fichiers à garder, on parle de la rétention

  — Vérifiez que la configuration est bien fonctionnelle.

  — Changez la configuration pour avoir un fichier de log/jour

  — Configurez également les noms de fichiers ; le fichier peut être nommé avec la date du jour.

7 Configuration de la timezone

— Vérifiez la date dans Cassandra. Que constatez-vous ?

— Lancez votre cqlsh directement avec la bonne TZ – sans modifier les fichiers de configuration

— Quittez la session et reconnectez-vous comme habituelle. Que remarquez-vous ?

— Modifiez l'affichage pour que la timezone du serveur concorde avec celle de votre système de manière. Cette modification est définitive(donc en modifiant le fichier de configuration nécessaire)

— Que remarquez-vous ?

Le cqlsh est codé en python, nous pouvons utiliser le module pytz pour afficher différentes timezone.

```
sudo pip install pytz
```

— Mettre la timezone CET

— Afficher la date sans la précision de la timezone

— Modifier le format de la date pour que l'on affiche la date sans la valeur de la timezone.