Projet de la matière:

SMB214: Réseaux ,systèmes embarqués et répartis

Sujet du projet : Apache Cassandra
Préparé par : Sanaa Allaw

Plan du projet :

Introduction au base de données NOSQL

I. Origine et concept du Cassandra
II. Principales caractéristiques
III. Architecture Cassandra
IV. Les cas d'utilisation
V. L'indexation dans Cassandra
VI. La répartition des données
VII. L'accès aux données
VIII. Communication entre les nœuds
IX. Securite dans appache cassandra
X. Installation et configuration
XI. Opération sur les keyspace est sur les table dans Cassandra

Conclusion

Introduction au NoSQL:

Le terme «NoSQL» a été inventé en 2009 lors d'un **événement** sur les bases de données distribuées.

-NoSQL désigne une catégorie de Systèmes de Gestion de Bases de Données (SGBD)

-Manipuler des bases de données dont les volumes sont très importants.

- En effet, NoSQL ne vient pas remplacer les BD relationnelles mais proposer une alternative ou compléter les fonctionnalités des SGBDR pour donner des solutions plus intéressantes dans certains contextes.

Mais pourquoi le NoSQL ?

- Cohérence : tous les noeuds du système voient exactement les mêmes données au même moment.
- Haute disponibilité (Availability) : en cas de panne, les données restent accessibles
- Tolérance au Partitionnement : Le système continue à fonctionner malgré la perte d'un message du à une panne. Autrement dit, en cas de morcellement du réseau, chaque sous-réseau doit pouvoir fonctionner de facon autonome.

Cassandra vs MySQL:

- Cassandra s'inscrit dans le mouvement NoSQL qui veut simplifier les bases de données en supprimant l'aspect relationnel.
  Les tables n'ont plus un schéma fixe prédéfini (que l'on peut en fait modifier ultérieurement), et peuvent évoluer horizontalement (quand aux colonnes) autant que verticalement (quand au lignes, donc aux enregistrements).

NoSQL signifie en fait Not Only SQL (pas seulement SQL), donc cela ne remet pas en cause le langage d'interrogation, qui est toujours SQL.

- Cassandra n'a pas de schéma et pas de table. Le nombre de colonnes peut varier d'une ligne à l'autre. MariaDB (alternative à MySQL) a implémenté un système de colonnes dynamiques qui permet de faire la même chose. Mais on sort du cadre SQL quand aux commandes.
- Voici un comparatif de performances fourni par Apache:
- Ecriture: MySQL: 300 ms. Cassandra: 0,12 ms.
- Lecture: MySQL: 350 ms. Cassandra: 15 ms.
- Différences de conception:
- Nombre de colonnes: MySQL: 4096. Cassandra: 2 milliards.
- Cassandra est moins fiable que MySQL et a une communauté et donc un support moins importants. On trouve beaucoup moins d'outils pour aider à le faire fonctionner, tels les interfaces graphiques, les gestionnaires comme PhpMyAdmin.
- Ce n'est pas un système comme MySQL qu'il suffit d'installer pour qu'il soit opérationnel, il convient de lire attentivement le manuel et d'adapter la configuration à votre projet logiciel (indépendamment de la création des tables).

Types du base de données NoSQL:

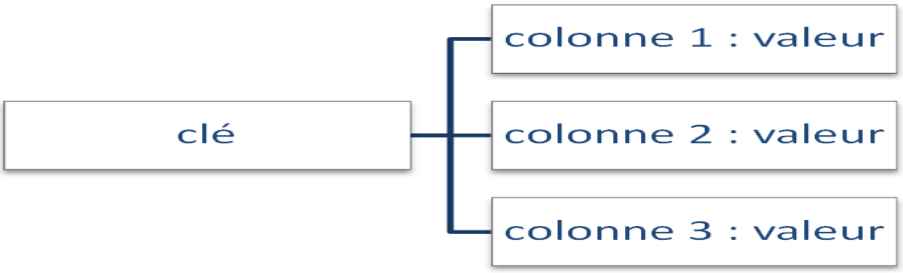
- Les solutions NoSQL existantes peuvent être regroupées en 4 grandes familles.

- Clé / valeur : Ce modèle peut être assimilé à une hashmap distribuée. Les données sont, donc, simplement représentées par un couple clé/valeur.

  Exemple :Amazon

- **Orienté colonne** : Ce modèle ressemble à première vue à une table dans un SGBDR à la différence qu'avec une BD NoSQL orientée colonne, le nombre de colonnes est dynamique. En effet, dans une table relationnelle, le nombre de colonnes est fixé .

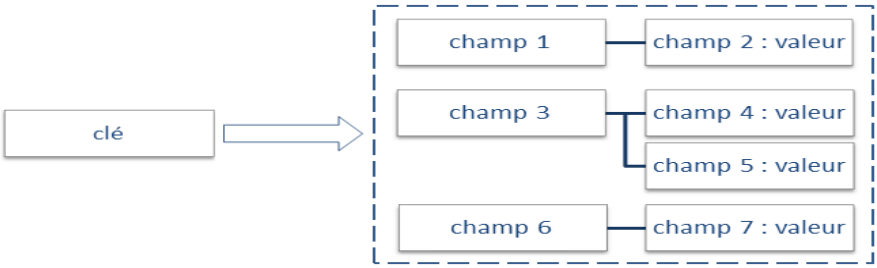
  Exemples: **Cassandra, HBase**



Orienté document : Ce modèle se base sur le form clé valeur. La valeur, dans ce cas, est un document de type JSON ou XML. L'avantage est de pouvoir récupérer, via une seule clé, un ensemble d'informations structurées de manière hiérarchique.

Cette représentation est considérée comme la plus adaptée au monde d'internet. Les données sont hiérarchisées

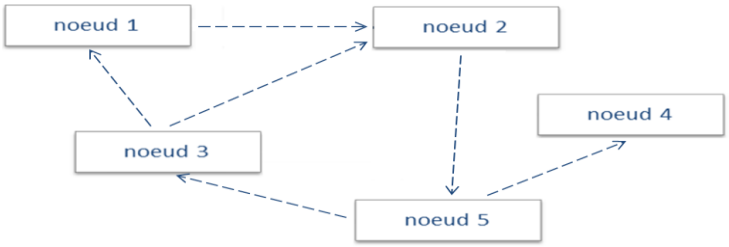
Exemples: **CouchDB, MongoDb**



Orienté graphe : Une base de données orientée graphe est une base de données orientée objet utilisant la théorie des graphes, donc avec des nœuds et des arcs, permettant de représenter et stocker les données .

On note :Dans le cas d'un réseau social par exemple, il peut être intéressant d'utiliser ce modèle, car il est plus facile retrouver une personne selon ses relations que selon son nom.

Exemples: **Neo4J, InfoGrid**



I.Origine et concept du Cassandra:

Initialement apache Cassandra a été développée en interne par Facebook puis a été diffusée en Open Source.

-Il est écrit en Java

- Utilise un protocole de sérialisation créé par facebook

-Cassandra est une base de données distribuée qui permet de stocker une grande quantité de données grâce à sa scalabilité horizontale

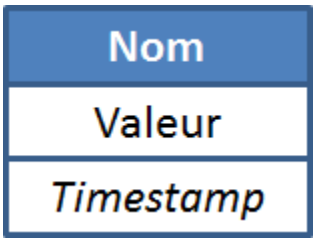
- Fournit un schéma de données dynamique afin d'offrir un maximum de flexibilité et de performance.

- Est une base de données NoSQL appartenant à la famille des bases de données orientées colonnes

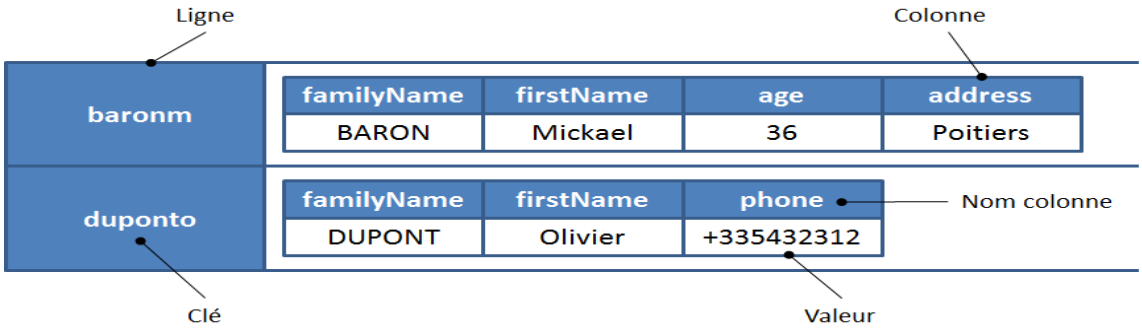
- Plusieurs grandes sociétés utilisent Cassandra pour leur application grand public. C'est le cas de Facebook, Twitter

- Mais pour bien comprendre cet outil, il faut tout d'abord bien assimiler le vocabulaire de base.

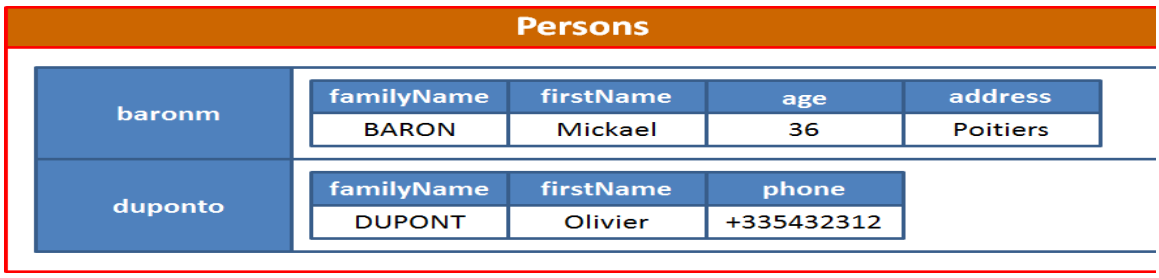
Colonne : Une colonne est la plus petite unité du modèle de données de Cassandra. C'est un triplet contenant un nom, une valeur et un timestamp. Ce dernier sert à déterminer la mise à jour la plus récente.



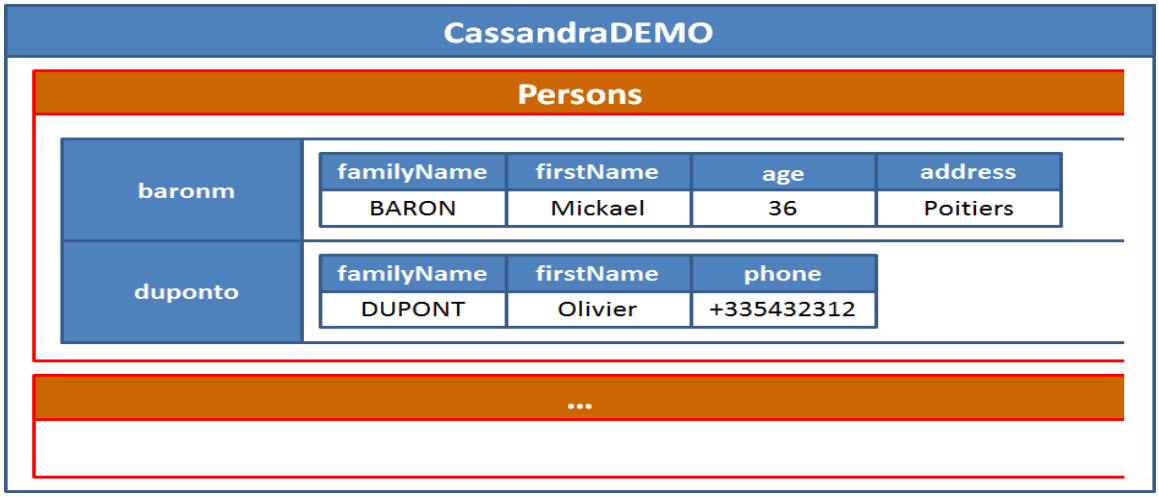
ligne : Une ligne est composée d'un ensemble de colonnes. Elle est identifiée par une clé.



Une famille de colonnes : ou *column family* en anglais est un regroupement logique de lignes. Pour faire le parallèle avec le monde des bases de données relationnelles, une famille de colonnes est en quelque sorte une table.



Keyspace :est un regroupement de famille de colonnes. Il s'agit d'une sorte de schéma si on compare au monde des bases de données relationnelles.



II.Cararectéristiques principales:

- Tolérance aux pannes : les données d'un nœud (un nœud est une instance de Cassandra) sont automatiquement répliquées vers d'autres nœuds (différentes machines). Ainsi, si un nœud est hors service les données présentes sont disponibles à travers d'autres nœuds. Le terme de facteur de réplication désigne le nombre de nœuds où la donnée est répliquée. Par ailleurs, l'architecture de Cassandra définit le terme de cluster comme étant un groupe d'au moins deux nœuds et un data center comme étant des clusters délocalisés. Cassandra permet d'assurer la réplication à travers

différents data center. Les nœuds qui sont tombés peuvent être remplacés sans indisponibilité du service.

- Décentralisé : dans un cluster tous les nœuds sont égaux. Il n'y pas de notion de maitre, ni d'esclave, le protocole GOSSIP utilisé pour découvrir la localisation et les informations sur l'état des nœuds d'un cluster.
- Élastique : lorsqu'un nouveau serveur est ajouté dans le cluster. Par ailleurs, Cassandra assure qu'il n'y aura pas d'indisponibilité du système ni d'interruption au niveau des applications.
- Haute disponibilité : possibilité de spécifier le niveau de cohérence concernant la lecture et l'écriture. On parle alors de *Consistency*. Apache Cassandra ne dispose pas de transaction. L'écriture des données est très rapide comparée au monde des bases de données relationnelles.

III.Architecture Cassandra:

- Une instance Cassandra est une collection de nœuds indépendants qui sont configurés ensemble pour former un cluster.
- Dans un cluster, tous les nœuds sont égaux, ce qui signifie qu'il n'y a pas de nœud maître ou un processus centralisant leur gestion.
- En fait, Cassandra utilise un protocole appelé Gossip afin de découvrir la localisation et les informations sur l'état des autres nœuds du cluster. Le protocole Gossip est un protocole de communication de type « *peer-to-peer »* dans lequel les nœuds échangent périodiquement des informations sur leur état mais également sur ce qu'ils savent des autres nœuds.
- Pour être plus précis, le processus s'exécute toutes les secondes afin d'échanger les messages avec au plus trois autres nœuds du cluster.

De plus, une version est associée à ces messages afin de permettre d'écraser les informations plus anciennes.

- Ainsi, quand un nœud démarre, il regarde dans son fichier de configuration les points de ralliement (SEED) qui devront au moins être contactés une fois.
- Cependant, afin d'éviter un partitionnement, tous les nœuds du cluster doivent disposer de la même liste de nœuds dans leur fichier de configuration.
- La détection des échecs est une méthode pour déterminer localement si un autre nœud est accessible ou pas. En outre, les informations récoltées par ce mécanisme permettent à Cassandra d'éviter d'émettre des requêtes aux nœuds qui ne sont plus accessibles.
- En fait, ce mécanisme fonctionne sur le principe de *heartbeat* soit de manière directe (en récoltant les informations directement des nœuds) soit de manière indirecte (en récoltant les informations par l'intermédiaire de la connaissance des autres nœuds).
- Lorsqu'un nœud est déclaré comme inaccessible, les autres nœuds stockent les messages susceptibles d'avoir été manqués par ce nœud. Cependant, il peut arriver qu'entre le moment où le nœud devient inaccessible et le moment où sa perte est détectée, un laps de temps s'écoule et qu'ainsi, les réplicas ne soient pas conservés. En outre, si le nœud vient à être indisponible pendant une période trop importante (par défaut, une heure), alors les messages ne sont plus stockés. C'est pour cette raison qu'il est conseillé d'exécuter régulièrement l'outil de réparation des données (NODE_REPAIR).

IV.Les cas d'utilisation de Cassandra:

- Bien que Cassandra soit dotée d'une architecture passionnante, supporte la scalabilité, la tolérance aux pannes, et bénéficie d'une haute disponibilité, ce n'est pas la meilleure solution pour tout type de problème. Voyons quels sont les cas d'utilisation les plus adaptés avec Cassandra.

I. **Systèmes distribués :**

Vous êtes sans doute comblés par l'architecture passionnante de Cassandra (réplication peer-to-peer, niveau de consistance, scalabilité, haute disponibilité, etc.). Aucune de ces qualités n'est vraiment nécessaire lorsque l'on souhaite installer une BD "single-node". Dans beaucoup de situations, une base de données installée sur une seule machine suffira. Par contre, si votre besoin est d'avoir une base de données distribuée sur plusieurs nœuds (même des dizaines), Cassandra est un des meilleurs choix.

II. **Beaucoup d'opérations d'écriture**

Dans une base de données relationnelle, l'écriture des données est coûteuse. Ceci est dû à la manière dont les données sont structurées : le SGBDR doit assurer l'intégrité des données dans les différentes tables au moment de chaque écriture. Cassandra a été conçue pour offrir une meilleure optimisation d'écriture de données. La stratégie d'écriture adoptée par Cassandra ainsi que son incompatibilité avec les transactions ACID permettent de donner plus de performance au niveau de l'écriture des données. *Une base de données qui fait beaucoup d'opérations d'écriture est plus performante avec Cassandra.*

III. **Données dynamiques, trop volumineuses**

Cassandra est une base de données orientée colonne, elle s'inspire du modèle *BigTable de google*. Sur chaque ligne on peut insérer un grand nombre de colonnes (jusqu'à deux milliards). Cassandra offre un plus par

rapport au modèle *BigTable*, car elle propose un concept de conteneur de colonnes ou *SuperColumns*. Le modèle orienté colonne offre une disposition différente du modèle relationnel. Les colonnes sur Cassandra sont dynamiques et peuvent changer d'une ligne à l'autre.

V.l'indexation dans Cassandra:

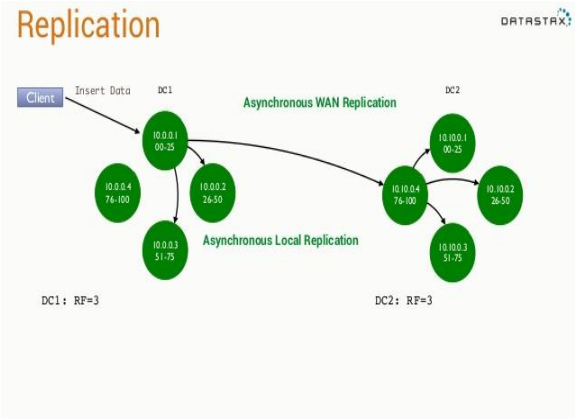
Un index est une structure de données qui permet un accès rapide ainsi qu'une recherche de données par rapport à un ensemble de critères donnés.

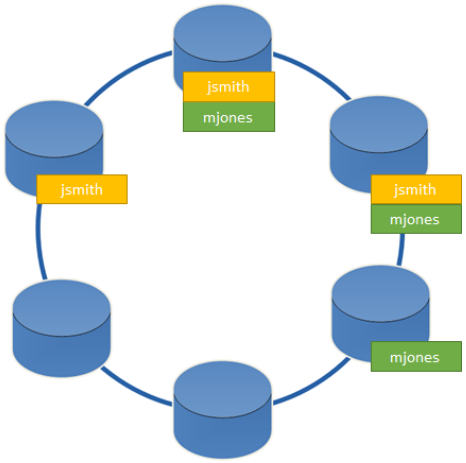
- Dans les bases de données classiques, une clé primaire est une clé unique utilisée pour identifier chaque ligne d'une table. Cela permet, comme tous les index, d'accélérer l'accès aux données dans une table.
- Dans Cassandra, l'index primaire d'une famille de colonnes correspond à l'index de ses clés de ligne. Puisque chaque nœud connait la plage de ses clés par nœuds gérés, les requêtes sur les lignes sont plus aisées à localiser en scannant l'index des lignes sur un nœud donné.

VI.La répartition des données:

- La réplication est le processus permettant de stocker des copies des données sur de multiples nœuds afin de permettre leur fiabilité et la tolérance à la panne. Quand un *keyspace* est créé dans Cassandra, il lui est affecté la stratégie de distribution des réplicas, c'est-à-dire le nombre de réplicas et la manière dont ils sont répliqués dans le cluster.
- La stratégie de réplication repose sur la configuration du cluster Snitch afin de déterminer la localisation physique des nœuds ainsi que leur proximité par rapport aux autres.

- Il est souvent fait référence au facteur de réplication (replication factor que nous nommerons RF par la suite) pour parler du nombre total de réplicas dans le cluster.
- Aussi :un facteur de réplication de 1 signifie qu'il n'y a qu'une seule copie de chaque ligne ;.
- un facteur de réplication de 2 signifie qu'il existe deux copies de chaque ligne .
- Tous les réplicas possèdent la même importance : il n'y a pas de réplicas principaux ou maîtres du point de vue de la lecture et de l'écriture.
- Ainsi, la règle générale est que le facteur de réplication ne doit pas être supérieur au nombre de nœuds dans le cluster. Cependant, il est possible d'augmenter le facteur de réplication puis d'ajouter le nombre de nœuds désirés à postériori. À noter toutefois que lorsque le facteur de réplication est supérieur au nombre de nœuds, alors les écritures ne sont plus prises en compte tandis que l'opération de lecture reste maintenue tant que le degré de consistance est respecté.
- Concernant les stratégies de distribution des réplicas, cela permet de jouer sur la façon dont les réplicas sont répartis dans le cluster pour un keyspace. Cette stratégie est à renseigner lors de la création de la keyspace.
- De plus, il est possible de configurer la manière dont les nœuds sont groupés ensemble dans la topology du réseau global. Cet élément de configuration correspond au snitch et est associé au cluster. Cassandra utilise alors cette information pour router les requêtes entre les nœuds.

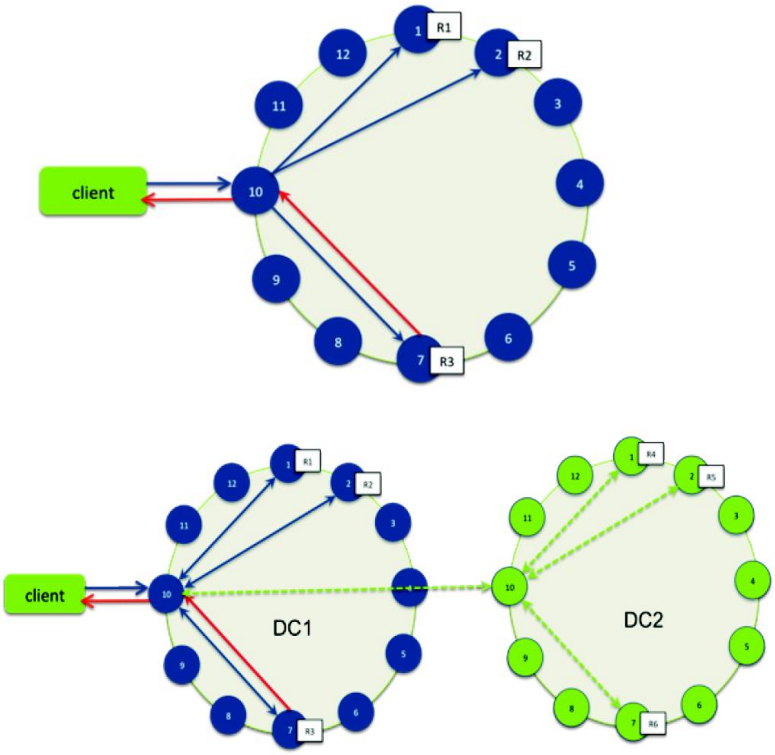




VII.L'accès aux données dans Cassandra:

- Tous les noeuds de Cassandra sont égaux. Ainsi, une demande de lecture ou d'écrire peut interroger indifféremment n'importe quel noeud du cluster. Quand un client se connecte à un noeud et demande une opération d'écriture ou de lecture, le noeud courant sert de coordinateur du point de vue du client.
- Le travail du coordinateur est de se comporter comme un proxy entre le client de l'application et les noeuds qui possèdent la données. C'est lui qui a la charge de déterminer quels noeuds de l'anneau devra recevoir la requête.

Cassandra privilégie toujours les accès séquentiels aux accès aléatoires, ce qui permet d'éviter une partie des latences importantes dues aux mécaniques des disques durs. Ainsi, lors d'une écriture, les données ne sont pas écrites directement sur disque mais stockées dans une table en mémoire ; un ajout dans un commitlog se comportant en append-only (et donc de manière séquentielle) permet d'assurer la durabilité de l'écriture. Lorsque la table en mémoire est pleine, elle est écrite sur le disque.



Concernant les requêtes d'écriture, le coordinateur émet la requête à tous les réplicas qui possèdent la ligne à modifier. Aussi longtemps qu'il sont disponibles, ils reçoivent les demandes d'écriture quelque soit le niveau de consistance demandé par le client. Le niveau de consistance d'écriture détermine le nombre de noeuds qui doivent acquitter l'écriture afin de considérer l'écriture comme ayant réussi.

Il est à noter que, dans le cas où il existe plusieurs *data center deployments*, Cassandra optimise les performances d'écriture en choisissant un noeud coordinateur dans chaque *data center* distant afin de traiter les requêtes des réplica dans le data center. Le noeud coordinateur contacté par l'application cliente n'a alors qu'à transmettre les requêtes d'écriture à un seul noeud de chaque *data center* distant.

Ecriture et lecture de data dans CASSANDRA

c'est faire intervenir 4 mécanismes principaux :

Ecriture de l’évènement : «Commitlog»,

Mise à disposition de la donnée le plus vite possible : «Memtable»,

Déclenchement du mécanisme qui rend la donnée consistante dans

le temps : «SSTable»,

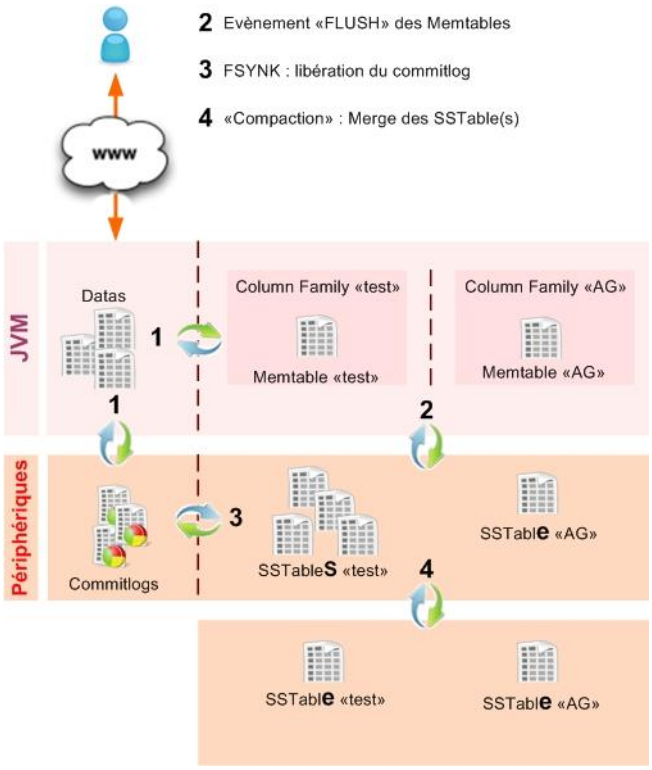
Ranger et Optimiser : «Compaction».

Opérations d'écriture

Toute activité d'écriture de noeuds est capturé par commit logs écrites dans les noeuds. Plus tard, les données seront capturées et stockées dans la mem-table. Chaque fois que le mem-table est pleine, les données seront écrites dans le fichier de données SStable. Toutes les écritures sont automatiquement partitionnées et répliquées dans le cluster. Cassandra consolide périodiquement le SSTable, en rejetant les données inutiles.

opérations de lecture

Pendant les opérations de lecture, Cassandra obtient des valeurs de la mem-table et vérifie le filtre bloom pour trouver la SSTable appropriée qui contient les données requises.



Composants de Cassandra

Les principales composantes de Cassandra sont les suivantes -

Node - Il est l'endroit où les données sont stockées.

Data center - Il est une collection de noeuds connexes.

Cluster - Une grappe est un composant qui contient un ou plusieurs centres de données.

Commit log - Le journal commettre est un mécanisme crash-reprise de Cassandra. Chaque opération d'écriture est écrit dans le journal de validation.

Mem table - Un mem-table est une structure de données résidant en mémoire. Après commettre journal, les données seront écrites dans le mem-table. Parfois, pour une famille mono-colonne, il y aura plusieurs mem-tables.

SSTable - Il est un fichier de disque sur lequel les données sont rincée de la mem-table lorsque son contenu atteint une valeur de seuil.

Bloom filtre - Ce ne sont que rapides, non déterministes, des algorithmes pour tester si un élément est un élément d'un ensemble. Il est un type spécial de cache. filtres de Bloom sont accessibles après chaque requête.

VIII.Communication entre les neouds dans Cassandra

Cassandra utilise un protocole appelé **Gossip** afin de découvrir la localisation et les information sur l'état des autres noeuds du cluster. Le protocole **Gossip** est un protocole de communication de type *peer-to-peer* dans lequel les noeuds échangent périodiquement des informations sur leur état mais également sur ce qu'ils savent des autres noeuds.

> Avec Gossip, les clients connaissent qui possède quoi. Il s' agit de garder l'information à jour entre les noeuds

2 types de flux:

1. Flux datas : enregistrement et lecture de la donnée,
2. Flux de services : communication entre les noeuds, gestion du partitionnement, stratégie de la réplication.

- Ainsi, quand un noeud démarre, il regarde, dans un premier temps, son fichier de configuration qui le renseigne sur le nom de son cluster ainsi que les autres noeuds compris dedans. Cela lui permet de les contacter afin de récupérer leur état.
- Cependant, afin d'éviter un partitionnement, tous les noeuds du cluster doivent disposer de la même liste de noeuds dans leurs fichiers de configuration.

- La détection des échecs est une méthode pour déterminer localement si un autre noeud est accessible ou pas. En outre, les informations récoltées par ce mécanisme permet à Cassandra d'éviter d'émettre des requêtes aux noeuds qui ne sont plus accessibles.

IX.Securité dans Appache cassandra :

Authentification interne

-Gestion des ID de login et des mots de passe dans la base de données.

Gestion de la permission des objects

-Contrôle d'accès aux objets et des actions des utilisateurs dans la base de données.

Encryption client a Noeud

-Protéger les données naviguant vers et depuis le cluster de la base de données.

-S'assure que les données ne peuvent pas être interceptées lors de l'acheminement au serveur.

IX.Installation de Cassandra sur Ubuntu:

Étape 1 - Installation de la machine virtuelle Java

Pour rendre le package Oracle JRE disponibles, vous devrez ajouter personnelles Archives emballage (PPA) en utilisant cette commande:

```
sudo add-apt-repository ppa:webupd8team/java
```

Mettre à jour la base de données du paquet:

sudo apt-get update

Ensuite, installez le Oracle JRE. L'installation de ce paquet particulier non seulement l'installe mais

aussi rend le JRE par défaut. Lorsque vous êtes invité, acceptez le contrat de licence:

sudo apt-get install oracle-java8-set-default

Après l'avoir installé, vérifiez qu'il est maintenant le JRE par défaut:

java –version

Vous devriez voir une sortie similaire à ce qui suit:

Output java version "1.8.0_60"Java SE Runtime Environment (build 1.8.0_60-b27) Java HotSpot 64-Bit Server VM (build 25.60-b23, mixed mode)

Étape 2 - Installation de Cassandra

- Installez les nouvelles mises à jour à l'aide des commandes suivantes

sudo apt-get update

sudo apt-get upgrade

- Ouverir /etc/apt/sources.list utilisant la commande suivante

sudo gedit /etc/apt/sources.list

- Ajouter les ligns suivant

deb http://www.apache.org/dist/cassandra/debian 10x main

deb-src http://www.apache.org/dist/cassandra/debian 10x main

- Inscrivez-vous et ajouter une touche de clés publiques et de mettre à jour à nouveau, notez que vous devrez peut-être modifier la clé conséquence

gpg --keyserver wwwkeys.pgp.net --recv-keys 4BD736A82B5C1B00

sudo apt-key add ~/.gnupg/pubring.gpg

sudo apt-get update

- Installez Cassandra utilisant la commande suivante

sudo apt-get install Cassandra

XI.Quelques Operations sur les keyspace dans Cassandra:

- Avant la version 0.8 de Cassandra, les développeurs Apache n'ont pas trouvé nécessaire d'avoir un langage d'interrogation de données comme SQL. Les clients voulant interroger Cassandra depuis une interface de ligne de commande utilisaient Cassandra-cli.
- Avec la sortie de la version 0.8, le langage d'interrogation de données pour Cassandra (CQL) est apparu, dont la syntaxe est basée sur SQL. Par contre, cela ne modifie pas le modèle Cassandra : il n'y a pas de support des jointures, de tri, ni de Group By.
- Ci-dessous quelques exemples de requêtes écrites avec CQL :

- cqlsh> CREATE KEYSPACE keyspace1 WITH strategy_class = 'org.apache.cassandra.locator.SimpleStrategy' AND strategy_options :replication_factor='1';
- cqlsh> USE keyspace1;
- Create d'un ColumnFamily cqlsh> CREATE TABLE USERS (USER_NAME varchar, PASSWORD varchar, AGE long, PRIMARY KEY (USER_NAME));
- Différences CQL/SQL

1. Pas de JOINTURES
2. Clause WHERE limitée
3. Clause ORDER BY limitée
4. Pas de contraintes

Un KEYSPACE appelé *cassandrademocql* via la commande *CREATE KEYSPACE*, en utilisant la stratégie de placement 'SimpleStrategy' et un facteur de réplication à 1.

SimpleStrategy

Utilisez uniquement pour un seul centre de données. SimpleStrategy place la première réplique sur un noeud déterminé par le programme de partitionnement

CQLSH

CREATE KEYSPACE cassandrademocql WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };

- **CQLSH**

  USE cassandrademocql

- **CQLSH**

- **ALTER KEYSPACE** cassandrademocql WITH strategy_class=SimpleStrategy AND strategy_options:replication_factor=2;

CQLSH

- **DROP KEYSPACE cassandrademocql**

Quelques Operation sur la table:

- **CREATE TABLE** emp (empID int, deptID int, first_name varchar, last_name varchar, **PRIMARY KEY** (empID, deptID));
- **INSERT INTO** Hollywood.NerdMovies (user_uuid, fan) **VALUES** (cfd66ccc-d857-4e90-b1e5-df98a3d40cd6, 'johndoe')
- **DROP TABLE** worldSeriesAttendees;

Conclusion:

- Cassandra est très rapide pour manipuler un volume important de données. Elle permet d'avoir des schémas de données flexible grâce à sa représentation en colonnes. De plus son architecture lui permet d'évoluer sans problème dans un environnement distribué, elle intègre des mécanismes de réplication de données et la possibilité de mettre en cluster plusieurs serveurs Cassandra.
- Pour ses accès disque, Cassandra privilégie toujours les accès séquentiels aux accès aléatoires, ce qui permet d'éviter une partie des latences importantes dues aux mécaniques des disques durs.
- Vous l'aurez compris, Cassandra est un formidable outil doté d'une architecture passionnante et pouvant servir de référence en architecture logicielle.

À utiliser, vous l'aurez compris, avec modération pour des besoins exceptionnels dans vos applications. Tout choix d'architecture logicielle est une question de compromis.