



Préparé par Cynthia Abou Maroun



Plan De Présentation

1. Introduction
2. Historique
3. Qui utilise Cassandra
4. Théorème CAP
5. Cassandra et CAP
6. Architecture de Cassandra
7. Structure et Indexage
8. CQL
9. CQL et Structure de données
10. Stockage
11. Distribution des données
12. Réplication des données
13. Connection Node
14. Requête Lecture/Ecriture
15. Application



Introduction

- Apache Cassandra est un Open Source (*NoSQL*) de base de données, décentralisée, et utilisée pour gérer une donnée massive soit structurée, demi-structurée ou non-structurée, sur des multiples de centre de donnée et le « Cloud ».
- Cassandra délivre une disponibilité continue, et orientée horizontalement, avec une évolution linéaire, et une simplicité opérationnelle sur plusieurs serveurs.
- Le modèle de données dans Cassandra offre la commodité d'index, avec un «log» de mise à jour structuré, avec un appui à la décentralisation de donnée, un « View » matérialisé et un cache intégré.



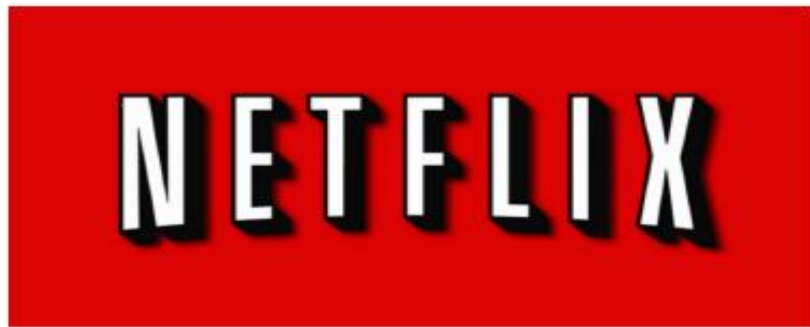
Historique

- Cassandra est fondée initialement par Facebook.
- En 2008 Facebook transmet ce projet à la fondation Apache.



Qui utilise Cassandra?

- Les Principaux Utilisateurs:







Théorème CAP

- Le théorème CAP ou CDP, aussi connu sous le nom de théorème de Brewer dit qu'il est impossible sur un système informatique de calcul distribué de garantir en même temps les trois contraintes suivantes, d'où on ne peut garantir que deux :
 - ✓ Consistency ou Cohérence (ou consistance des données): tous les nœuds du système voient exactement les mêmes données au même moment;
 - ✓ Availability ou Disponibilité: garantie que toutes les requêtes reçoivent une réponse;
 - ✓ Partition-Tolerance ou Tolérance au Partitionnement: aucune panne moins importante qu'une coupure totale du réseau ne doit empêcher le système de répondre correctement (ou encore: en cas de morcellement en sous-réseaux, chacun doit pouvoir fonctionner de manière autonome).



Cassandra et CAP

- Cassandra est généralement classé comme un système AP (Disponible et Tolérant)
- Mais Cassandra peut être réglé avec le facteur de réplication et niveau de la cohérence pour répondre aussi au C (Cohérence).



Architecture de Cassandra

- Cassandra est une base de données distribuée sur plusieurs serveurs appelés nœuds. Les données sont distribuées entre ces nœuds.
- Ces nœuds sont tous égaux : pas de notions maître/esclave.
- La communication entre eux pour la gestion des données.
- L'ensemble des nœuds est appelé un cluster.
- Chaque nœud est un serveur.
- Le processus s'exécute toutes les secondes.
- Chaque nœud a un fichier de configuration où ils stockent les informations concernant les autres.



Structure et Indexage

- Pour travailler sur Cassandra il faut savoir les termes suivants : colonnes, lignes, famille de colonnes et keyspace.
 - Une **colonne** est la plus petite unité de modèle dans Cassandra. Elle est constituée d'un nom, d'une valeur et d'un timestamp. Le timestamp détermine quand cette colonne a été créée et par suite savoir la valeur la plus récente. Le nom peut atteindre 64Koets, la valeur 2Goets.

Nom
Valeur
<i>Timestamp</i>

- Une **ligne** est un ensemble de colonnes. Elle peut contenir jusqu'à deux milliards de colonnes. Elle possède une clé qui peut atteindre aussi 64 Koctets. On remarque que les colonnes ne sont pas identiques entre les lignes.

The diagram shows two rows of data. The first row is labeled 'baronm' and contains four columns: 'familyName' (BARON), 'firstName' (Mickael), 'age' (36), and 'address' (Poitiers). The second row is labeled 'duponto' and contains three columns: 'familyName' (DUPONT), 'firstName' (Olivier), and 'phone' (+335432312). Labels with arrows point to various parts: 'Ligne' points to the 'baronm' row header; 'Colonne' points to the 'address' column header; 'Nom colonne' points to the 'phone' column header; 'Clé' points to the 'duponto' row header; and 'Valeur' points to the value '+335432312'.

baronm	familyName	firstName	age	address
	BARON	Mickael	36	Poitiers
duponto	familyName	firstName	phone	
	DUPONT	Olivier	+335432312	

- La **famille de colonnes** est un ensemble de lignes. Elle correspond à peu près à une table dans une base de données ordinaire.
- Le **Keyspace** est un ensemble de famille de colonnes.

Persons				
baronm	familyName	firstName	age	address
	BARON	Mickael	36	Poitiers
duponto	familyName	firstName	phone	
	DUPONT	Olivier	+335432312	



CQL?

Le langage de Cassandra s'appelle CQL.

- CQL se compose de déclarations. Comme SQL, les déclarations permet de changer, de chercher et de stocker des données, ou simplement de modifier la façon dont les données sont stockées.
- Les déclarations se terminent par un point-virgule (;).
- Par exemple, ce qui suit est la syntaxe CQL valide:

```
cqlsh:demo> SELECT * FROM users WHERE lastname='Smith';
```

- Pour plus d'information sur CQL:
http://docs.datastax.com/en/cql/3.1/cql/cql_reference/cqlCommandsTOC.html



CQL et Structure de données

L'API recommandée pour créer des schémas de Cassandra depuis 0,7 est via CQL. Mais Cassandra encourage développeur à partager les informations de schéma.

Pourquoi? Parce que même si CQL ressemble beaucoup à SQL, ils ne fonctionnent pas d'une manière similaire à l'intérieur.

Rappelez-vous, pour chaque famille de colonne, ne pensez pas à une table relationnelle. Au lieu de cela, pensez à une structure de données trié et imbriqué.



CQL et Structure de données

- Exemple:

```
INSERT INTO example (field1, field2, field3) VALUES (1,2,3);  
INSERT INTO example (field1, field2, field3) VALUES (4,5,6);  
INSERT INTO example (field1, field2, field3) VALUES (7,8,9);
```

- Les données sont stockées de la manière suivante:

```
RowKey: 1  
=> (column=, value=, timestamp=1374546754299000)  
=> (column=field2, value=00000002, timestamp=1374546754299000)  
=> (column=field3, value=00000003, timestamp=1374546754299000)  
-----  
RowKey: 4  
=> (column=, value=, timestamp=1374546757815000)  
=> (column=field2, value=00000005, timestamp=1374546757815000)  
=> (column=field3, value=00000006, timestamp=1374546757815000)  
-----  
RowKey: 7  
=> (column=, value=, timestamp=1374546761055000)  
=> (column=field2, value=00000008, timestamp=1374546761055000)|  
=> (column=field3, value=00000009, timestamp=1374546761055000)
```



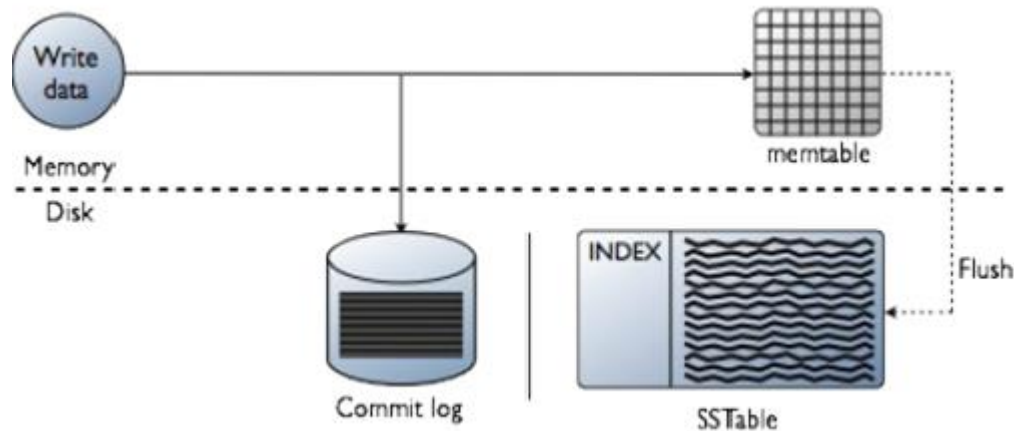

CQL et Structure de données

- Pour chaque exemple ci-dessus, il y a 3 choses importantes à regarder: la clé de ligne (RowKey: <?>), Le nom de colonne (<?> Colonne =) et la valeur de la colonne (valeur = <?>). A partir de ces exemples, nous pouvons faire quelques observations initiales au sujet de la cartographie (mapping) des états CQL à leurs représentations internes.
- Vous avez peut-être aussi remarqué que ces lignes contiennent tous colonnes sans nom de colonne et aucune valeur de colonne. Ceci n'est pas un BUG! Il est en fait une façon de gérer le fait qu'il devrait être possible de déclarer l'existence de field1 = <un nombre> sans préciser nécessairement des valeurs pour le field2 ou field3.



Où sont les données stockées?

- Pour chaque famille de colonne, il y a 3 couche de stockage de données: memtable, commit log et SSTable.
- Pour plus d'efficacité, Cassandra ne répète pas les noms des colonnes dans la mémoire ou dans le SSTable.

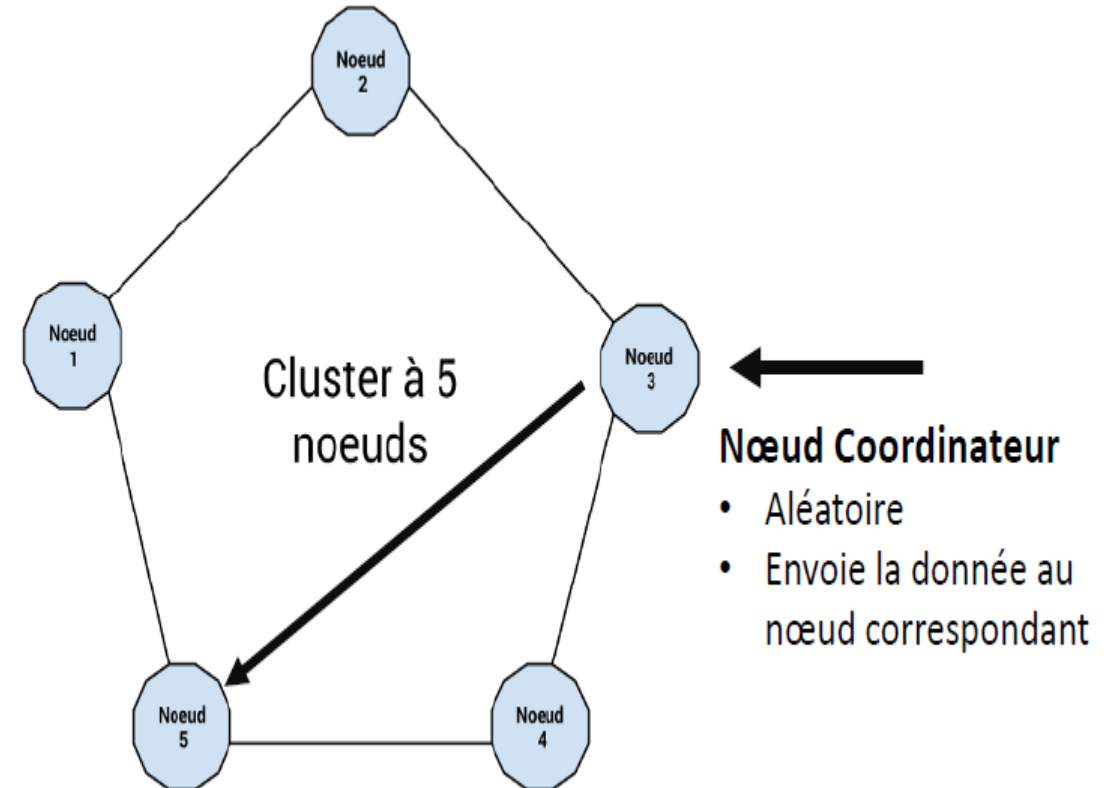




Distribution des données


- De manière ordonnée, chaque nœud prend en charge une plage de clé de partition triée par ordre croissant.
- De manière aléatoire, chaque nœud prend en charge une plage de la clé de partition distribuée uniformément. D'où:

- ✓ Cle de partition: K
- ✓ Fonction de Hachage(ex:MD5): F()
- ✓ Token = F(K)
- ✓ Token $\in [0 - (2^{127} - 1)]$
- ✓ MD5 $\rightarrow [0 - (2^{127} - 1)]$





Réplication de données

- Cassandra est conçu comme un système peer-to-peer qui fait une copie des données et distribue les copies au sein d'un groupe de nœuds.
- Cassandra utilise le mécanisme de hachage ([Hash](#)) cohérente pour distribuer des données sur un cluster. Chaque groupe a un partitionneur configuré pour le hachage de chaque clé de partition.
- Pour une bonne tolérance aux pannes, Cassandra réplique ses données sur les nœuds de cluster.
- Le nombre de replication (= nombre de noeuds) Facteur de replication (RF)
- Si RF = 1 cad une seule replication (Replicas) , RF =2 cad 2  répliquations



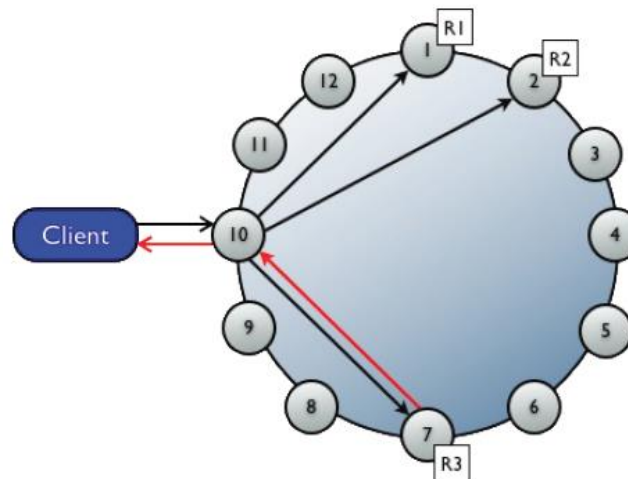
Connection Node

- Le noeud de connexion dépend de la configuration du pilote de côté client.
- Par exemple, si vous utilisez les pilotes «DataStax», il y a deux configurations principale à côté client qui contrôle quelle nœud il faut parler à:
 - **Points de contact**: est une liste d'un ou plusieurs adresse de nœud. Lors de la création d'une instance de cluster à côté client, le conducteur tente de se connecter aux nœuds spécifiés dans les "points de contact" dans l'ordre. Si elle ne parvient pas à se connecter au premier nœud, essayez la prochaine ... Comme pourvu que l'un nœud est connecté avec succès, il ne tente pas de se connecter plus les autres nœuds.
 - **Politiques d'équilibrage de charge**: Par défaut, une instance de cluster côté client gère les connexions à tous les nœuds du cluster et connecte au hasard un nœud pour toute demande du client, ce qui pourrait ne pas être assez performant, surtout quand vous avez plusieurs centres de données. «Les politiques d'équilibrage de charge » sur une instance de cluster détermine la stratégie d'attribution de connexion des nœuds aux demandes des clients.



Requête Lecture/Ecriture

- Tous les nœuds de Cassandra sont égaux. Ainsi, une demande de lecture ou d'écrire peut interroger indifféremment n'importe quel nœud du cluster. Quand un client se connecte à un nœud et demande une opération d'écriture ou de lecture, le nœud 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 nœuds qui possèdent la donnée. C'est lui qui a la charge de déterminer quels nœuds de l'anneau devront recevoir la requête.
- Si le niveau de consistance choisi est **ONE** ou **LOCAL_QUORUM**, alors seuls les nœuds du même « data center » que le nœud coordinateur doivent acquitter l'écriture.



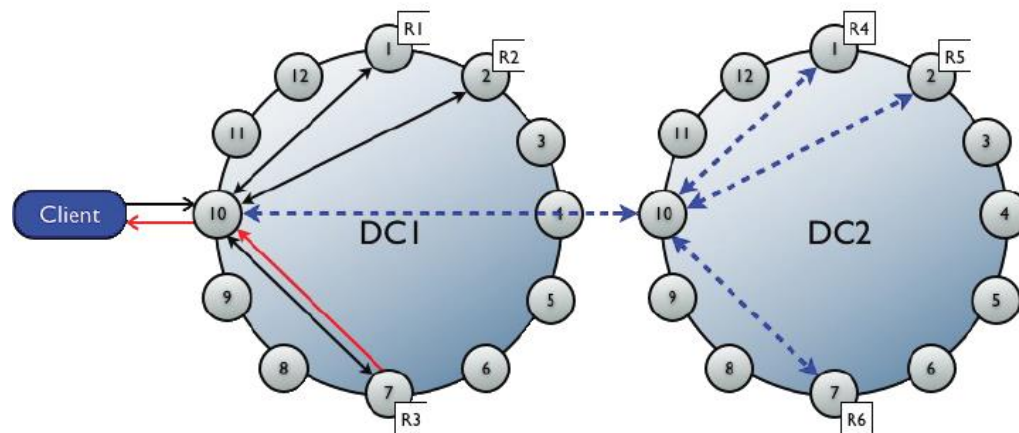
- On distingue, ici, trois types de cohérence en écriture / lecture (et on a à choisir un):
 - ✓ 1. **ALL** : le cluster attend la réponse de tous les nœuds avant d'écrire ou lire la donnée avant de répondre au client,
 - ✓ 2. **ONE** : le cluster attend seulement la réponse d'un des nœuds avant de répondre au client,
 - ✓ 3. **QUORUM** : le quorum est défini comme la majorité absolue des répliques. Si la donnée est répliquée 3 fois, alors le cluster attendra la réponse de 2 nœuds avant de répondre au client, c.à.d. $\text{quorum} = (\text{nombre de réplication} / 2) + 1$.

NB : Si on choisit le type 'ALL', dans ce cas, on peut avoir la cohérence absolue. Cassandra, ici, perd la disponibilité puisqu'il faut s'assurer à chaque fois que l'opération a été faite partout, ce qui rend le système beaucoup moins disponible.



Requête Ecriture

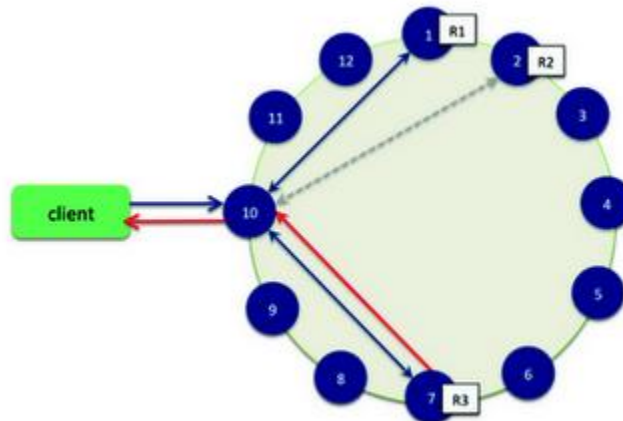
- Concernant les requêtes d'écriture, le coordinateur émet la requête à tous les réplicas qui possèdent la ligne à modifier. Aussi longtemps qu'ils sont disponibles, ils reçoivent les demandes d'écriture quel que soit le niveau de consistance demandé par le client. Le niveau de « consistance » d'écriture détermine le nombre de nœuds qui doivent acquitter l'écriture afin de considérer l'écriture comme ayant réussi.
- Dans le cas où il existe plusieurs data center « deployments », Cassandra optimise les performances d'écriture en choisissant un nœud coordinateur dans chaque data center distant afin de traiter les requêtes des réplicas dans le data center. Le nœud coordinateur contacté par l'application cliente n'a alors qu'à transmettre les requêtes d'écriture à un seul nœud de chaque data center distant.





Requête Lecture

- Concernant la lecture, il y a deux types de requêtes de lecture qu'un coordinateur peut émettre à un réplica :
 - ✓ **une requête de lecture directe.** Dans ce cas, le nombre de réplicas contactés par une demande de lecture directe est déterminé par le niveau de consistance spécifié par le client ;
 - ✓ **une requête de réparation de lecture en tâche de fond.** Dans ce cas, elle est envoyée à tous les réplicas additionnels qui n'ont pas reçu de requête directe. Ce type de requête permet de vérifier que la ligne est consistante par rapport aux autres réplicas.





Installation



Application



References

- <https://www.infoq.com/fr/articles/modele-stockage-physique-cassandra>
- http://blog.xebia.fr/wpcontent/uploads/2015/07/Data_Xebia_Cassandra_Programmez.pdf
- <http://soat.developpez.com/articles/cassandra/#LI>
- <http://mbaron.developpez.com/tutoriels/nosql/cassandra/installation-outilsadministration/>
- <https://hostpresto.com/community/tutorials/how-to-install-apache-cassandra-on-ubuntu-14-04/>
- <https://www.digitalcean.com/community/tutorials/how-to-run-a-multi-node-cluster-database-with-Cassandra-on-Ubuntu-14-04>



Merci pour votre attention