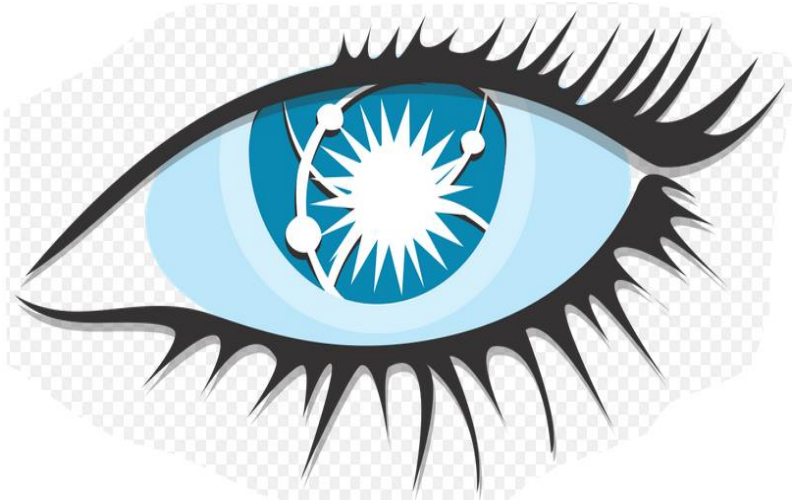

Cassandra

Cassandra



Formation
CASSANDRA

- Introduction - Rappel
- Cassandra architecture…
- La réplication des données
- La cohérence des données
- Chemins de lecture & écriture
- Mécanique de maintenance
- Cassandra & Spark

- Introduction - Rappel
- Cassandra architecture…
- La réplication des données
- La cohérence des données
- Chemins de lecture & écriture
- Mécanique de maintenace
- Cassandra & Spark

Qu’est ce que le Big Data

Descriptions des systèmes big data:

Applications concernées par les 3 ‘V’:

- Volume: des GigaBytes aux TéraBytes et au-delà
- Vitesse: les capteurs de données, le flux de ‘clic’, les transactions financières..
- Variété: les données doivent pouvoir être ingérées depuis plusieurs formats

Caratéristiques requises:

- Disponibilité multi-régions
- Réponse très rapide et fiable
- Pas de maillon faible (SPOF)

Pourquoi pas de base relationnelle ?

Le modèle de données relationnel fournit:

- Des tables normalisées, sans redondances d'information
- Des tables dont les informations sont liées par jointure
- Une conformité au modèle A.C.I.D (Atomicité, Cohérence, Isolation, Durabilité)
  - A : t**ous les changements apportés aux données sont effectués totalement** ou pas du tout (commit,rollback)
  - C: les transactions doivent **respecter les contraintes d'intégrité des données de la base de données (C.I.F)**
  - I: **les écritures et lectures des transactions réussies ne seront pas affectées** par les écritures et lectures d'autres transactions
  - D: **transactions réussies survivront de façon permanente** et ne seront pas affectées par d'éventuelles pannes ou problèmes techniques. (redolog)
- Mais, tout cela a un coût très élevé :
  - Les jointures sur des milliards de lignes nécessite beaucoup de puissance
  - La ventilation des tables sur différents serveur est complexe et fragile (RAC)
- Les applications modernes ont différentes priorités:
  - Besoin de vitesse et de disponibilité prioritairement à la cohérence immédiate
  - De nombreux serveurs 'low cost 'en rack plutôt que des systèmes centralisés très couteux
  - Le besoin en garantie transactionnelle temps réel est limité

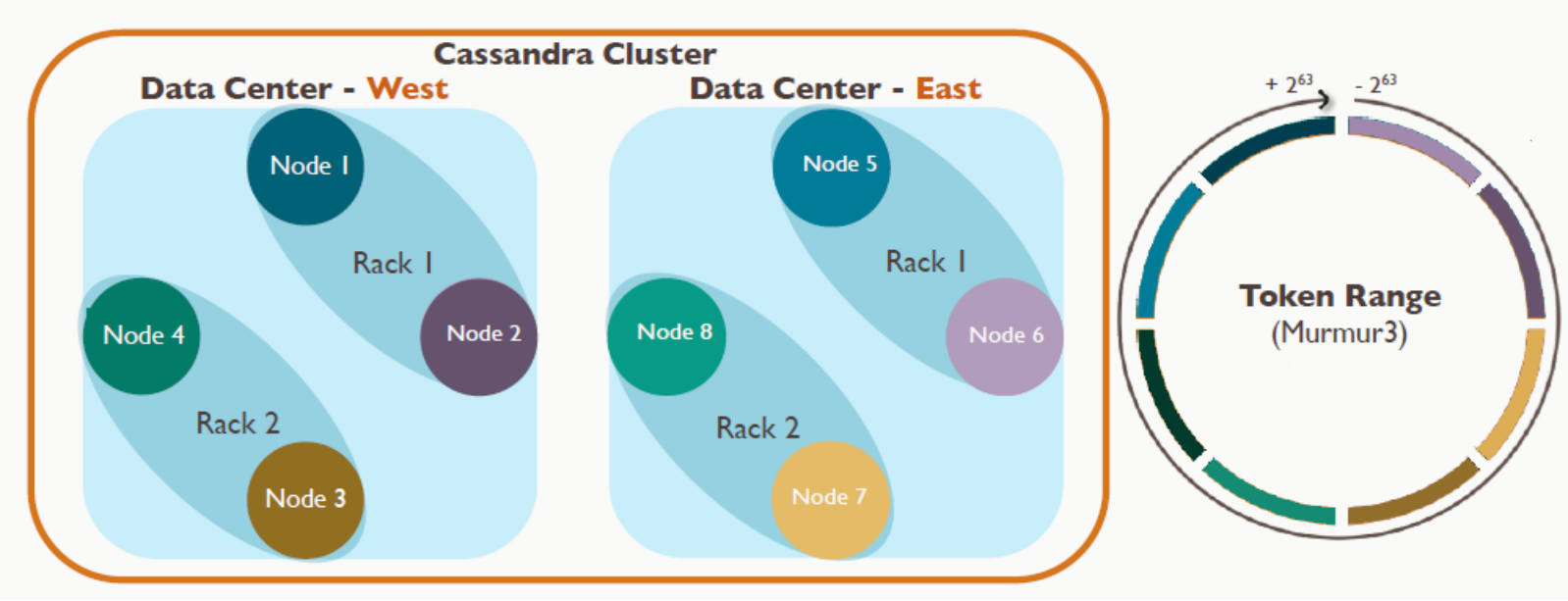
Quelles stratégies aident à gérer le ‘big data’

Distribution des données à travers des nœuds

Assouplissement du niveau de consistance des données

Assouplissement dans la définition des schémas

Optimisation des modèle de données pour répondre aux besoins



Le paysage NOSQL:

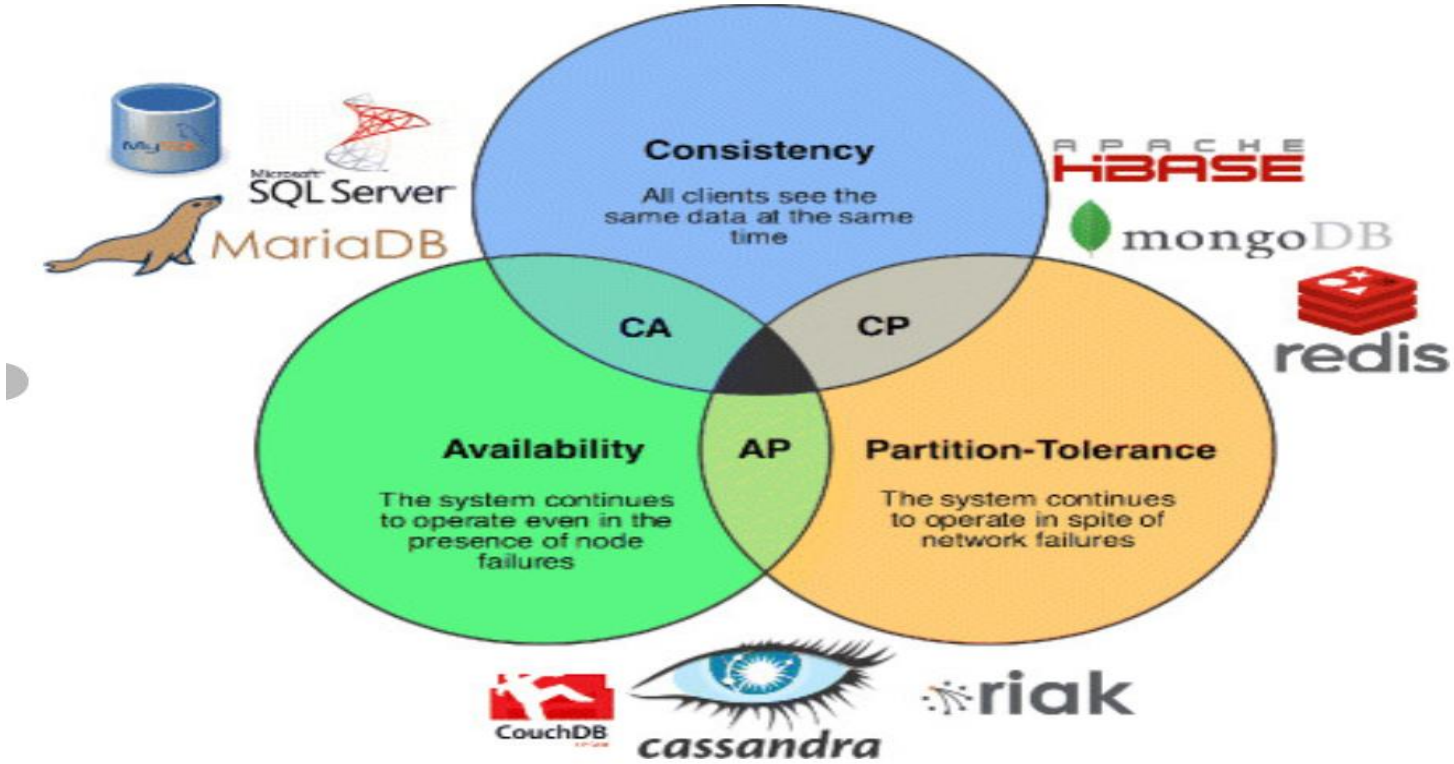
Quatre larges classes de bdd non-relationnelles:

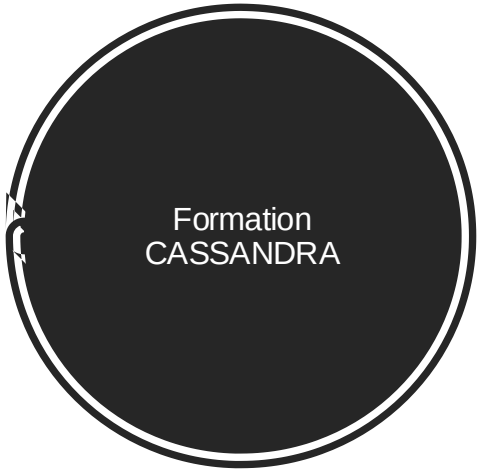
- Graph: les données sont reliées à n autres dans une logique de graphe (ou de réseau)
- Clé-Valeurs: les clés correspondent à des valeurs arbitraires de tout type
- Document: des ensembles de documents interrogeables en entier ou par partie
- Colonne: les clés correspondent à des ensemble de colonne typées

Trois facteurs clés nous permettent d'évoluer dans ce paysage:

- Consistance: obtenez vous le même résultats quelques soient le nœud interrogé ?
- Disponibilité: est ce que le cluster répond lors de très forte sollicitations en lecture ou écriture?
- Perte de Partition: est ce que le cluster est toujours accessible quand une partie dysfonctionne?

Qu’est ce que le théorème de C.A.P?





Introduction - Rappel

Cassandra architecture…

La réplication des données

La cohérence des données

Chemins de lecture & écriture

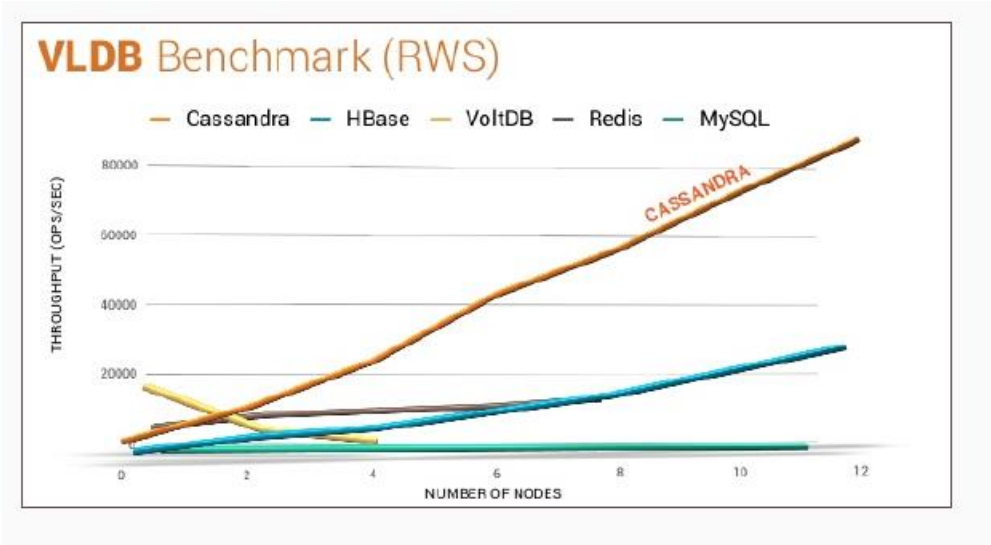
Mécanique de maintenace

Cassandra & Spark

Qu’est ce que Cassandra?

Une base de données scalable linéairement:

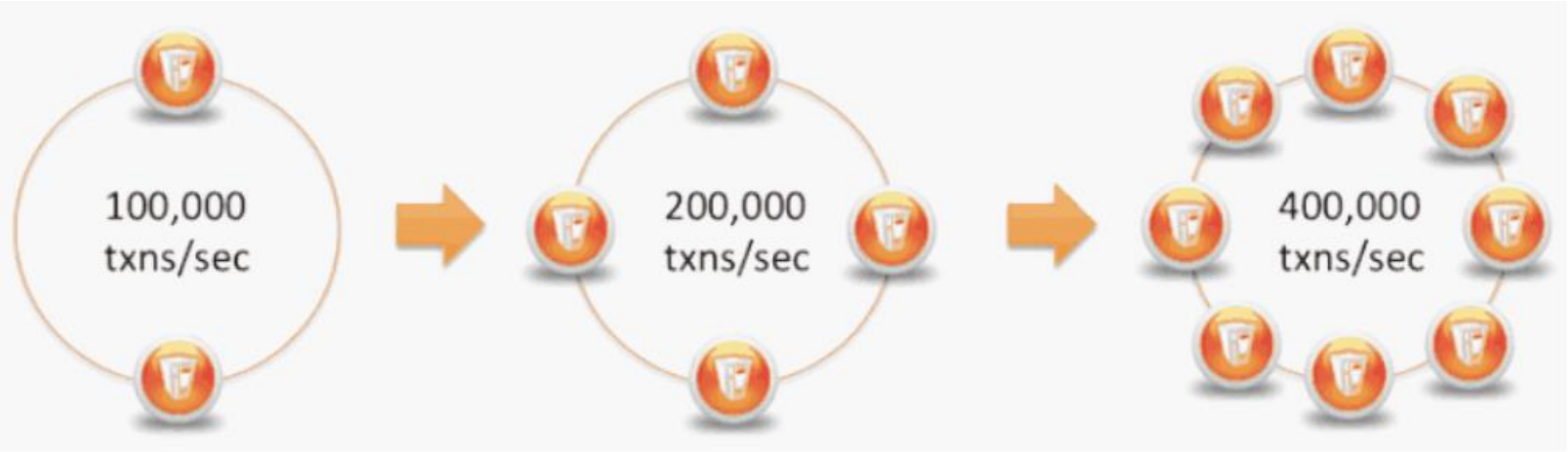
- Entièrement distribuée sans SPOF
- Gratuite et open source, avec une communauté importante de développeur
- Très performante, avec une scalabilité quasi linéaire avec des cadres d’usages adaptés



Qu’est ce que Cassandra?

Pas de SPOF grâce à la scalabilité horizontale:

- Scabilité horizontale: on ajoute des nœuds au cluster
- Scalabilité verticale: on ajoute de la RAM et de la CPU au serveur



Comment Cassandra a évolué?

Au cœur de sa technologie nous avons:

- Google Big Table: les fondations du modèle de stockage
- Amazon Dynamo DB: les fondations du modèle de système distribué
- Facebook: intégre Big Table et Dynamo DB pour lancer Cassandra

Quand Cassandra est le meilleur choix de base de données

Cassandra est excellent lorsque vous avez besoin de :

- Pas de SPOF
- Ecritures en temps réel avec de l'analyse opérationnelle dynamique
- Un modèle de données souple et facilement adaptable
- Une scalabilité horizontale quasiment linéaire avec des serveurs lambda
- Une réplication fiable des données dans différents datacenters
- Des schéma de tables clairement définies

Quand Cassandra n’est pas le bon choix?

Les RDBMs sont parfaits lorsque le besoin est:

- Des transactions ACID, avec possibilités de rollbacks (par exemple des transferts inter-bancaire)

Quels sont les cadres d’usages courants de Cassandra?

Cassandra est particulièrement utiles pour :

- Gérer les playlists et les collections (comme Spotify par exemple)
- Les moteurs de personnalisation et de recommandation (comme Ebay par ex)
- Les messageries (comme Instagram par ex)
- La détection de fraude
- Qui utilise Cassandra:
  - Netflix, Twitter, Call of Duty, Expedia, Nasa, Cisco, Ebay, Orange avec PNS, etc..

Comprendre l’architecture interne de Cassandra

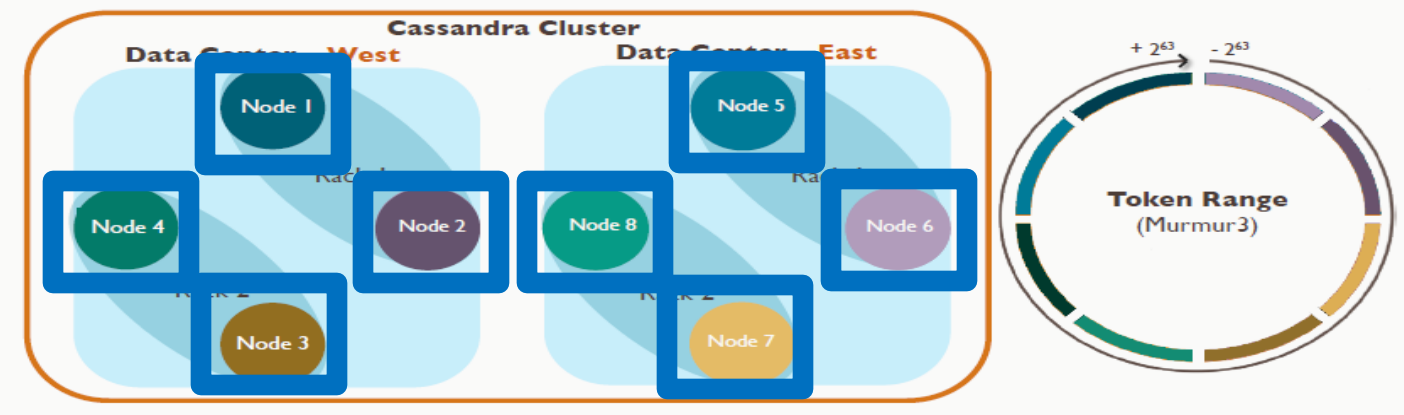
Objectifs d’apprentissage:

- Comprendre comment les requêtes sont coordonnées
- Comprendre régler la consistance
- Introduction aux opérations d’anti-entropies
- Comprendre la communication inter-nœuds
- Comprendre le keyspace ‘system’

Qu’est ce qu’un cluster? (1)

Un ensemble de nœuds en ‘peer to peer’ (P2P)

- Node : une instance Cassandra
- Rack: un ensemble logique de nodes
- Data Center: un ensemble logique de racks
- Cluster: un ensemble complet de nodes qui décrit intégralement un anneau de ‘token’

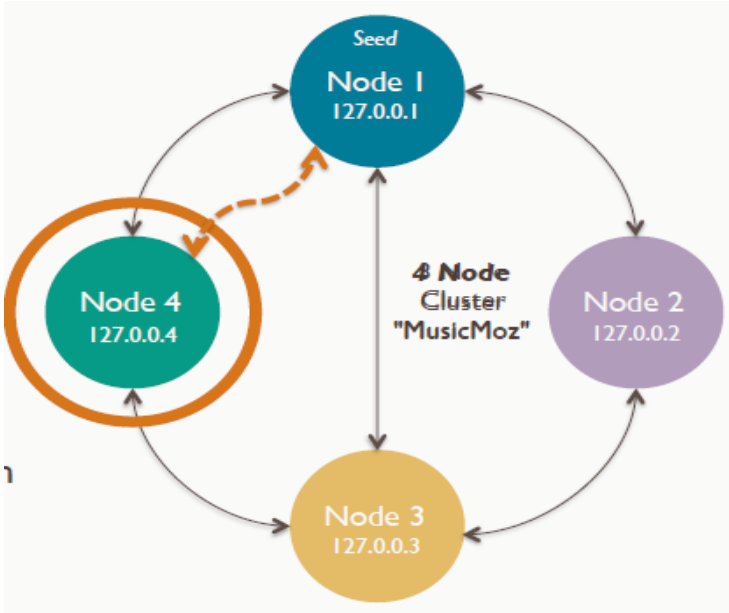


Qu’est ce qu’un cluster? (2)

Les nodes rejoignent un cluster à partir de leur propre fichier de configuration ‘cassandra.yaml’

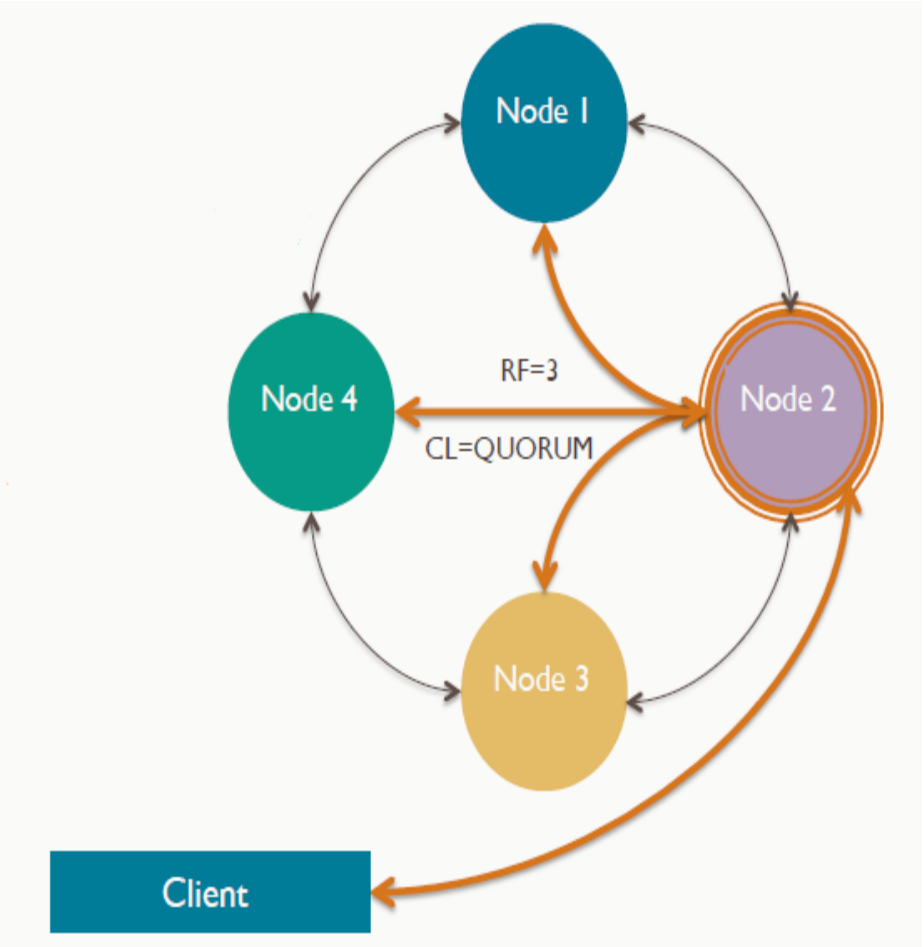
Les principaux paramètres de configuration sont :

- Cluster_name: nom du cluster Cassandra
- Seeds: adresse IP des nœuds d’initialisation qui permettrons aux nouveaux nœuds de s’intégrer au cluster en découvrant sa topologie (2 par DC est une bonne pratique)
- Listen_adress: adresse IP par laquelle ce nœud communiquera



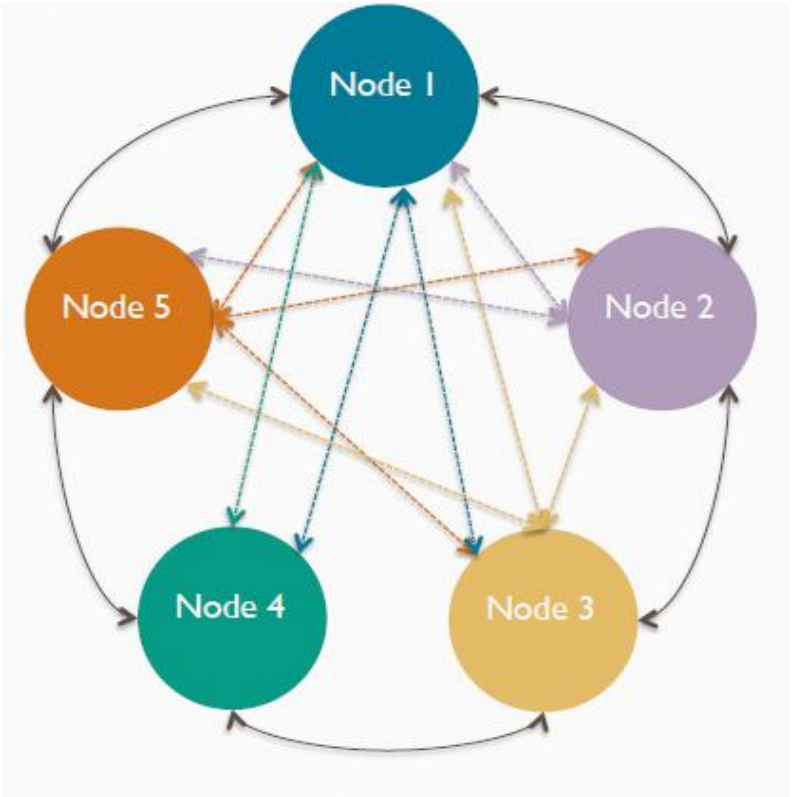
Qu’est ce que le coordinateur?

- C’est le nœud choisi par le client (driver) pour recevoir une requête en lecture ou écriture
  - N’importe quel nœud peut coordonner n’importe quelle requête.
  - Chaque requête de client peut être coordonnées par un nœud différent.
- Pas de S.P.O.F
  - C’est un principe fondamental de l’architecture Cassandra



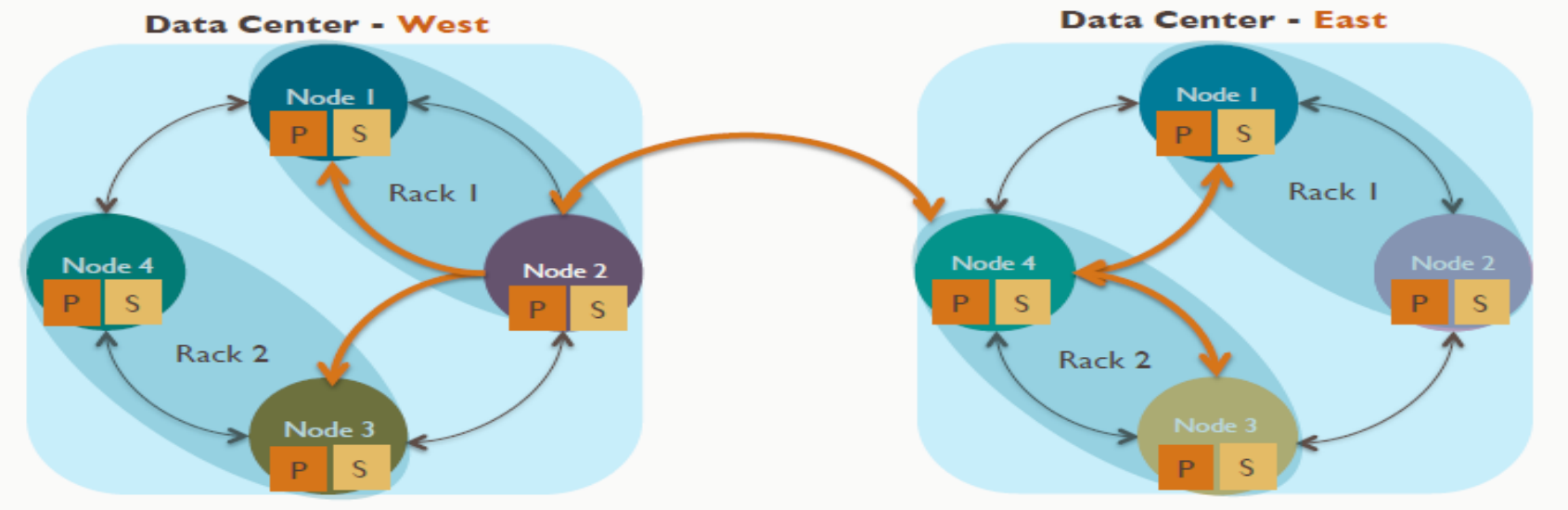
Comprendre comment les nœuds communiquent:

- Qu'est ce que le protocole Gossip?
  - Une fois par seconde chaque nœud contacte de 1 à 3 autres nœuds afin d'échanger des informations:
    - Sur le status des nœuds (heartbeats)
    - Sur la localisation des nœuds
    - Ces échanges et acquittements sont horodatés donc continuellement mise à jour et abandonnées.
- Lorsqu'un nœud rejoint le cluster il 'Gossip' avec les nodes déclarés 'seed' afin de découvrir la topologie du cluster
  - Les 'seeds' : sont déclarés dans le 'cassandra.yaml'
  - La liste des seeds est la même pour chaque DC
  - Lorsqu'il y a plusieurs DC , il faut croiser un seed dans la liste des seeds de chaque DC



Qu’est que le Snitch et comment le configurer? (1)

- Le Snitch :
  - Informe son partitionner de la topologie des Racks et des DCs
  - Permet au partionner de ventiler les réplicas en veillant qu’au sein d’un même DC tous les réplicas ne soient pas sur le même rack (socle physique), ce qui pourrait engendrer une perte de données en cas de perte d’un rack; et cela par datacenter.



Qu'est que le Snitch et comment le co

- Encore une fois cela se fait par le fichier cassandra.
  - Avec le paramètre endpoint_snitch qui peut pren

- SimpleSnitch – node proxim… keyspace, single data … only
- PropertyFileSnitch – node proximity … configuration in cassandra-topology.…
- GossipingPropertyFileSnitch – node … and data center … sandra-rackdc.…
- YamlFileNet…orkTopologySnitch – n… data center …
- RackInferringSnitch –
- Ec2Snitch – Amazon E… and EC2 Availability Z…
- Ec2MultiRegionSnitch … broadcast_address, en…
- GoogleCloudSnitch –
- CloudstackSnitch – fo…

Property File Snitch

- Reads datacenter and rack information for all nodes from a file
- You must keep files in sync with all nodes in the cluster
- cassandra-topology.properties file

```
175.56.12.105=DC1:RAC1
175.50.13.200=DC1:RAC1
175.54.35.197=DC1:RAC2
175.54.35.152=DC1:RAC2

120.53.24.101=DC2:RAC1
120.55.16.200=DC2:RAC1
120.57.18.103=DC2:RAC2
120.57.18.177=DC2:RAC2
```

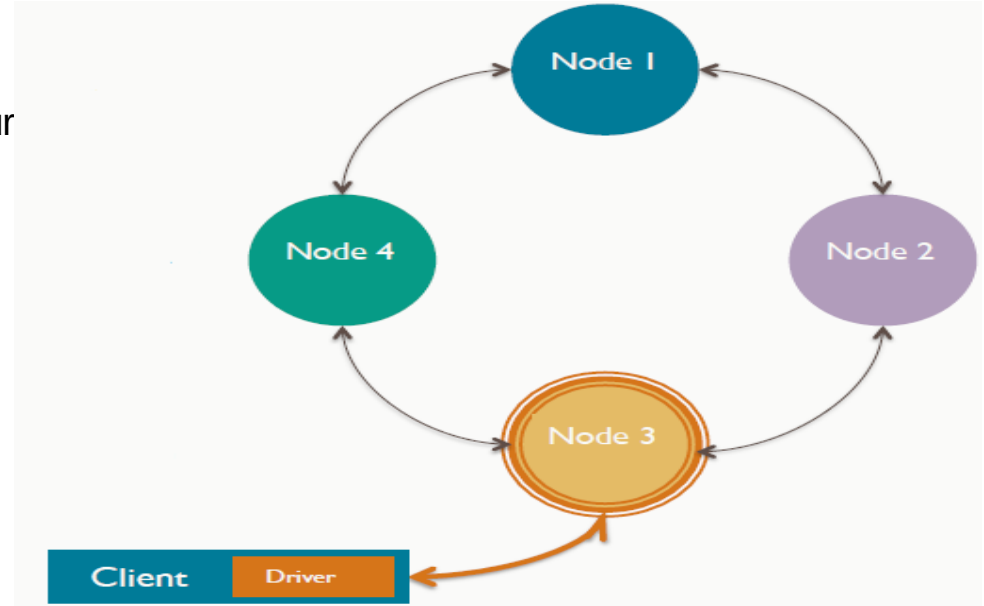
Gossiping Property File Snitch

- Relieves the pain of the property file snitch
- Declare the current node's DC/rack information in a file
- You must set each individual node's settings
- But you don't have to copy settings as with property file snitch
- Gossip spreads the setting through the cluster
- cassandra-rackdc.properties file

```
dc=DC1
rack=RAC1
```

Comment les requêtes clientes sont coordonnées? (1)

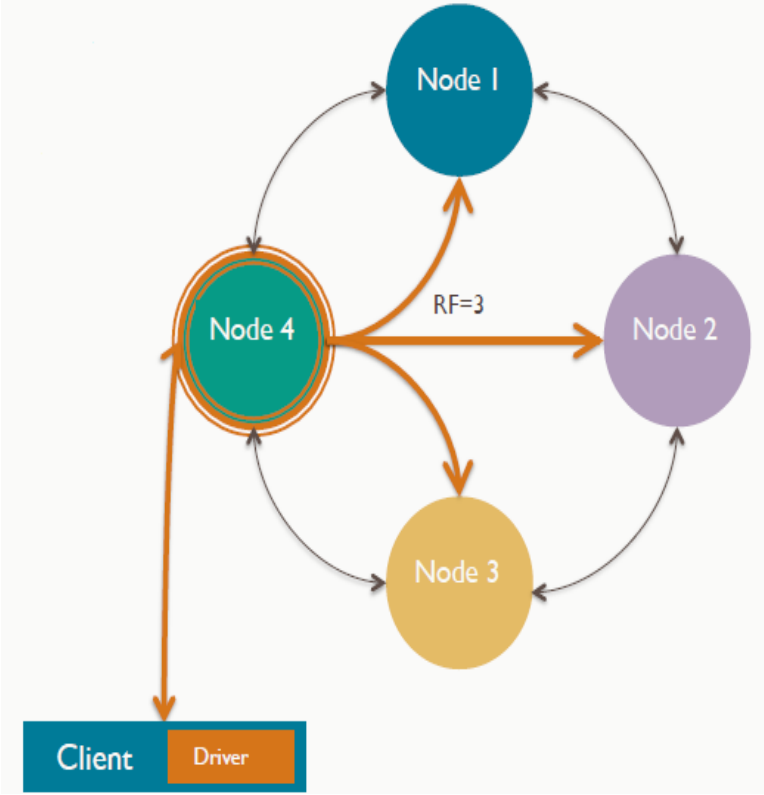
- Le driver C* choisit le nœud vers lequel toutes les requêtes seront envoyées
  - Le driver client fournit une API pour gérer les requêtes
  - La politique par défaut du driver est :
    - TokenAware DCAWARE
  - Datastax maintient les drivers open source pour
    - Java, Python et C principalement
  - La commaunauté C* maintient les drivers pour
    - PHP, Perl, Go, R et Scala principalement



Comment les requêtes clientes sont coordonnées? (2)

- Le coordinateur gère les coefficients de réplication (RF):
  - Réplication factor (RF) : sur combien de nœud une écriture doit elle être copiée?
  - Plage de valeurs possible de 1 au nombre de nœud du cluster.
  - Le RF est positionné par keyspace et par DataCenter.

- Chaque écriture vers chaque nœud est individuellement horodatée.

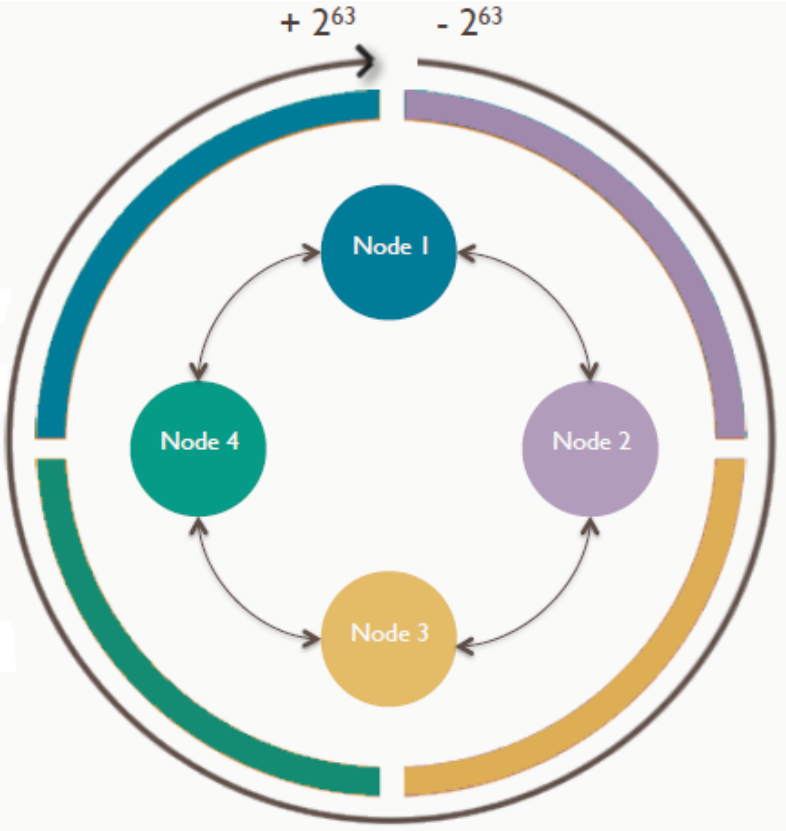


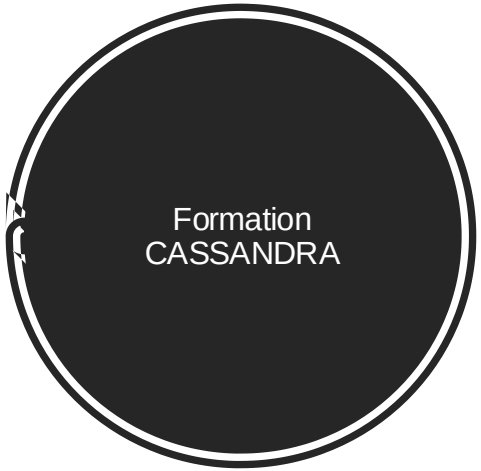
Comment les requêtes clientes sont coordonnées? (3)

- Le coordinateur applique aussi le niveau de consistance (CL)
  - Consistency Level (CL): combien de nœuds doivent acquitter la lecture ou écriture d'une requête
  - Le CL peut changer d'une requête à l'autre
  - En cas de succès le coordinateur notifie le client
- Les principaux niveaux de consistance possible sont :
  - ANY
  - ONE
  - QUORUM (RF/2)+1
  - ALL
  - …

Qu’est ce que le hashage consistent?

- Les données sont stockées dans les nœuds sous forme de partitions; identifiée par un numéro de token unique
  - Partition: est un espace de stockage propre à chaque nœud (analogie possible avec une ligne d’une table)
  - Token: valeur entière générée par un algorithme de hashage qui identifie la localisation de la partition dans le cluster C*
- Les 2^64 valeurs qui servent aux intervalles de token d’un cluster sont utilisées comme un seul anneau (ring)
  - donc , toute partition est localisable à partir d’un ensemble consistant de valeur de hashage, indifféremment du nœud qui l’héberge
  - Les intervalles de HV dépendent du choix du partitionner





- Introduction - Rappel
- Cassandra architecture…
- La réplication des données
- La cohérence des données
- Chemins de lecture & écriture
- Mécanique de maintenace
- Cassandra & Spark

Réplication des données

Comprendre la réplication

- Comment les nœuds sont organisés en rack et datacenter?
  - Un cluster de nœuds peut être logiquement organisé en racks et datacenters
    - Node: le serveur physique ou virtuel d'une seule instance Cassandra
    - Rack: un regroupement logique de nœuds physiques
    - DataCenter: un regroupement logique d'un jeu de racks
- Permet de connaitre le cheminement en écriture et lecture des requêtes
  - La topologie du cluster est communiquée par les processes Snitch et Gossip
- Chaque nœud appartient a un rack d'un datacenter
- L'identité de chaque rack de nœuds et datacenter doit être configurée dans le fichier
  - Cassandra-rackdc.properties
    - dc=DC1
    - Rack=RAC1

Comment le keyspace impacte la réplication?

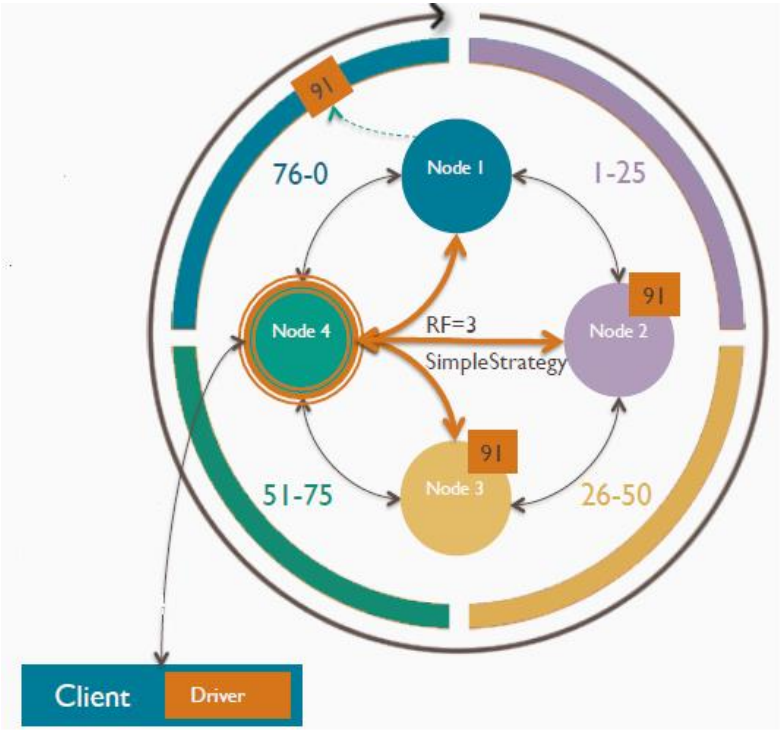
- Le facteur de réplication est configuré lors de la création du keyspace

  - SimpleStrategy (juste pour l'apprentissage): un seul paramètre pour tout le cluster
    - nommé : 'replication_factor'
    - Exemple :
      - CREATE KEYSPACE simple-demo
        WITH REPLICATION = {'class':'SimpleStrategy', 'replication_factor':2}

  - NetWorkTopologyStrategy : dans ce cas on utilise un paramètre pour chaque DC du cluster
    - Nom de datacenter affecté dans le fichier cassandra-rackdc.properties
    - Exemple:
      - CREATE KEYSPACE simple-demo
        WITH REPLICATION = {'class':'NetworkTopologyStrategy', 'dc-east':'2','dc-west':3}

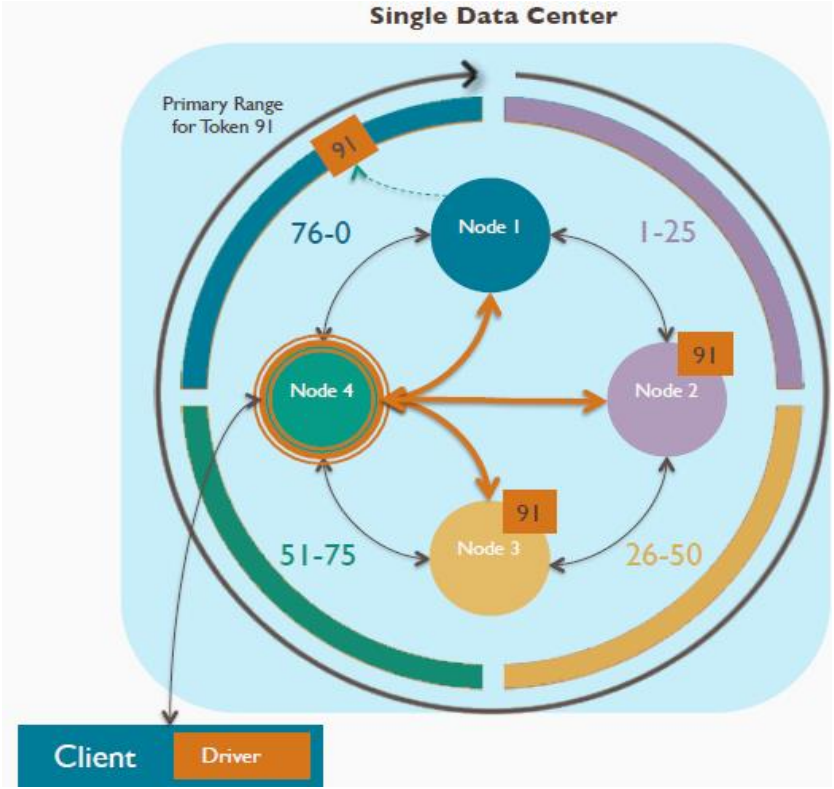
Comment le coordinateur transfert les demandes d'écritures?

- Le keyspace de la table cible détermine:
  - Facteur de réplication : combien de réplicas à faire pour chaque clé de partition
  - La stratégie de réplication: sur quels nœuds doit on placer ses réplicas
- Toutes les partitions sont des réplicas, ils n'y a pas de partitions 'originelles'. Un peu de terminologie:
  - Premier réplicas : placé sur le nœud qui possède un segment de token primaire qui contient le token cible
  - Le nœud le plus proche (Closest Node): réplicas placé dans le même rack et datacenter (si possible)
  - Subsequent replicas (si RF>1) : placés dans le segment secondaire des autres nœuds.



Comment les données sont répliquées entre les noeuds?

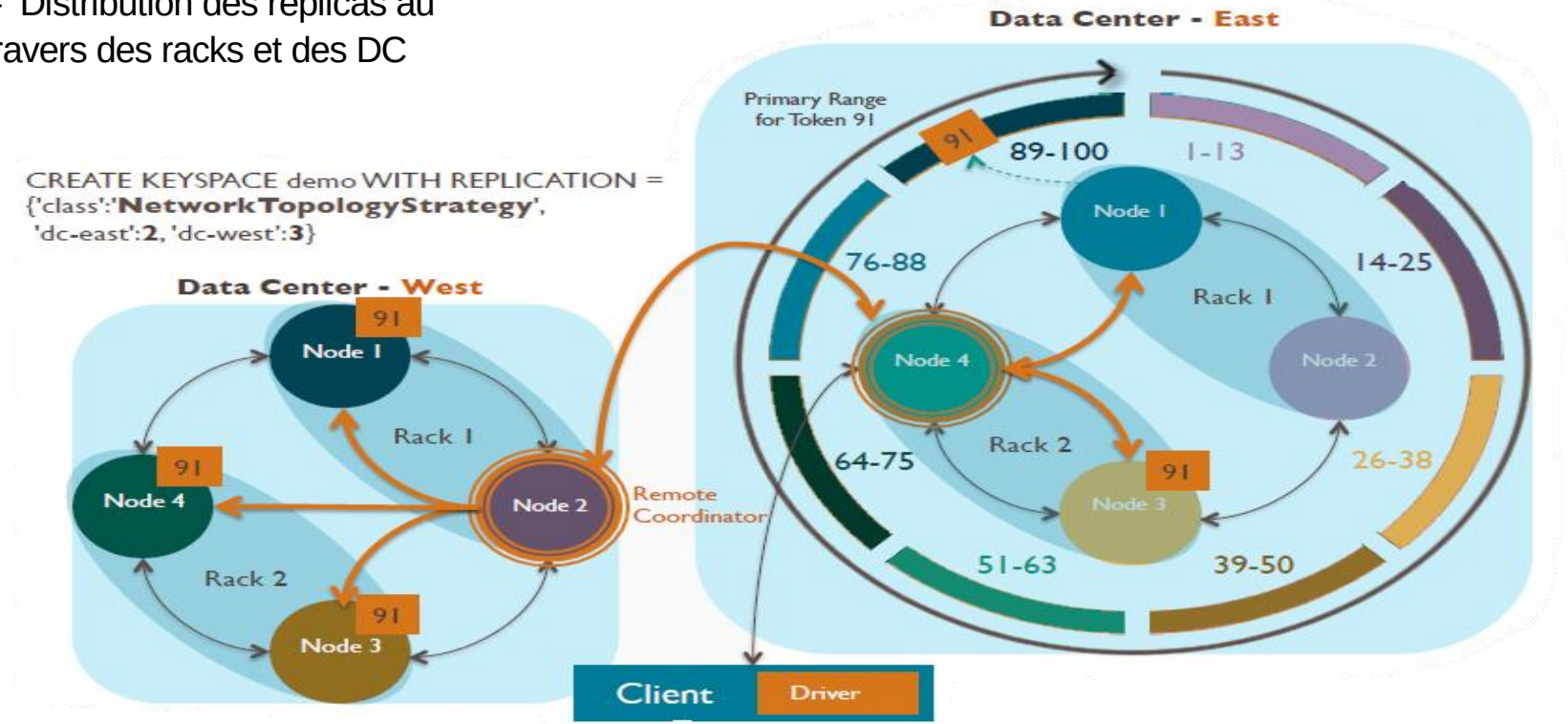
- SimpleStrategy: créer les réplicas sur les nœuds voisins au nœud propriétaire du segment primaire
  - CREATE KEYSPACE demo with replication= {‘class’:’SimpleStrategy’,’replication_factor’:3}
  - RF=3 est une recommandation minimale



Comment les données sont répliquées entre les DataCenter?

NetworkTopologyStrategy:

– Distribution des réplicas au travers des racks et des DC

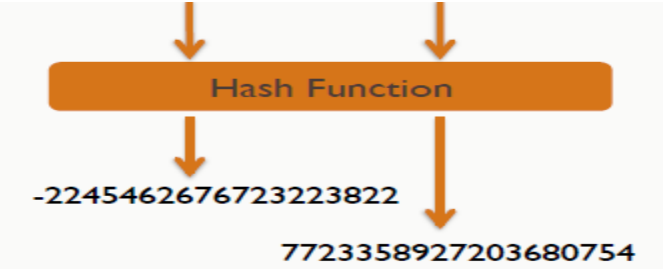


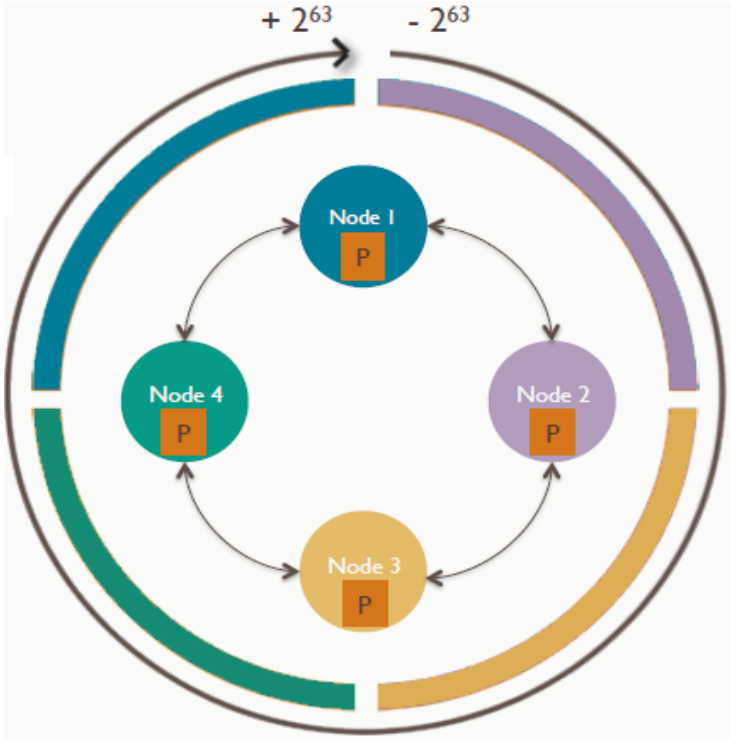
Partitionnement

Qu’est qu’un partitionner? (1)

- Un système existant sur chaque nœud qui génère des token à partir des variables qui lui sont fournis lors d’insertion de lignes
  - Function de hashage : convertie une variable de longueur variable en une valeur donnée de longueur fixe

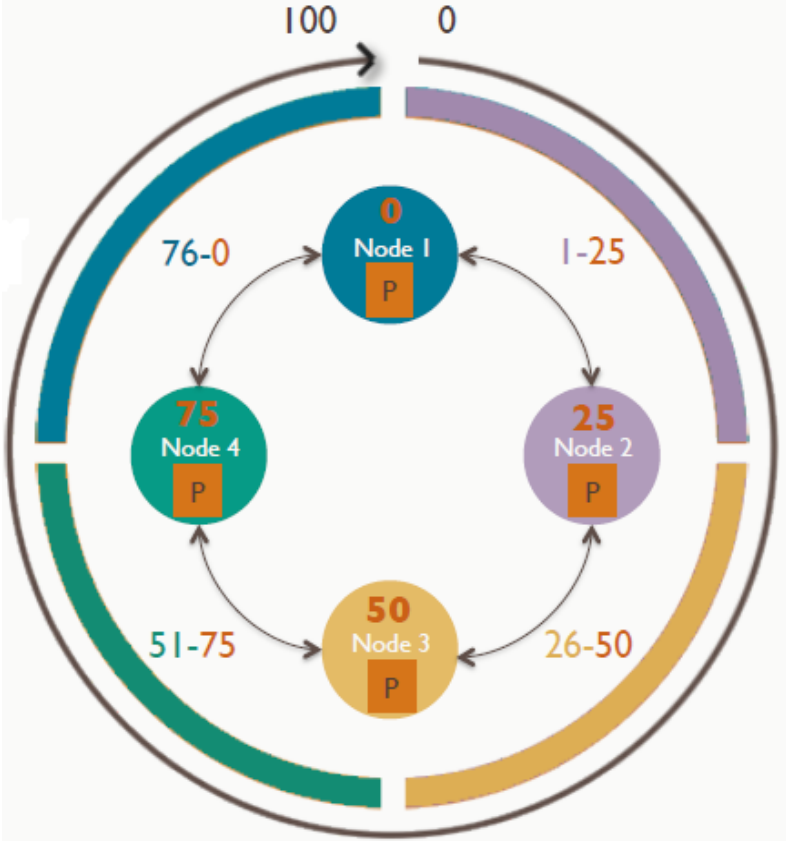
« @97203





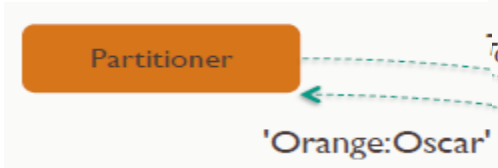
Qu’est qu’un partitionner? (2)

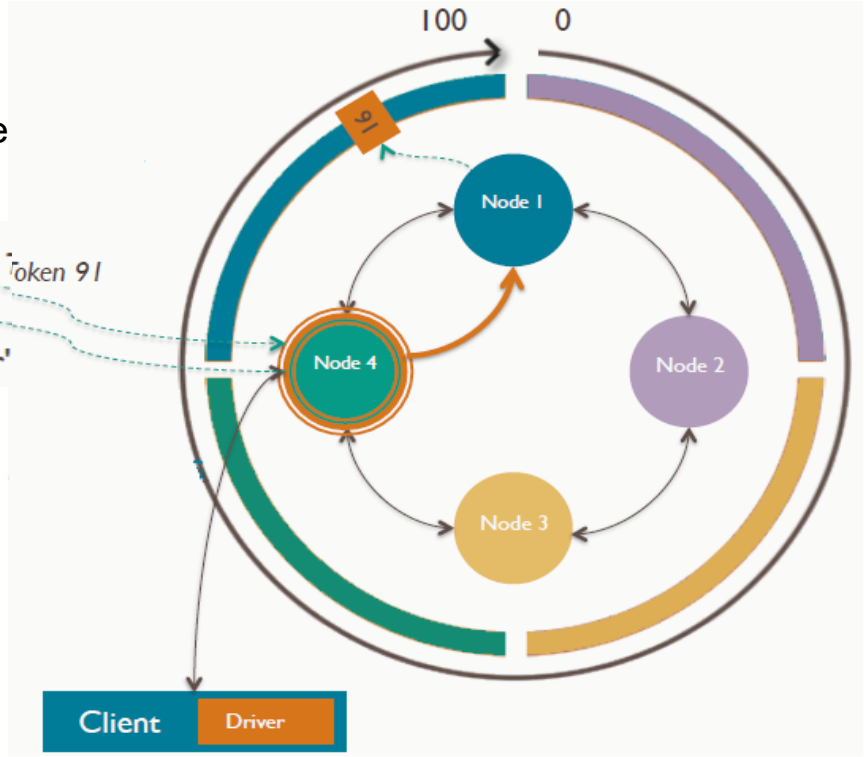
- Prenons par exemple un anneau allant de 0 à 100 tokens (au lieu de l’intervalle -2^63 à 2^63)
  - Chaque nœud se voit assigner un token comme chacune de ses partitions
  - Le token d’un noeud est la plus grande valeur de son intervalle de valeur dédié
- Ce segment de plage de token est qualifié de primaire par rapport aux autres segment de token hébergé sur le même nœud
  - Le nœud peut aussi héberger des réplicas en fonction du RF. On parle alors de ‘secondary range’.



Comment fonctionne le partitionner?

- Le partitionner d'un node coordinateur, choisit par le driver client, génère un token à partir de la clé de partition d'une requête en écriture
  - Le premier réplica est envoyé au nœud responsable du token





- La clé primaire d'une table détermine la valeur de sa 'clé de partition', comme dessous:
  - Create table Users (firstname text, lastname text, level int, PRIMARY KEY((lastname,firstname));
  - Insert into users (firstname,lastname,level) values ('Oscar','Orange',42)

Quels sont les différents types de partitionner?

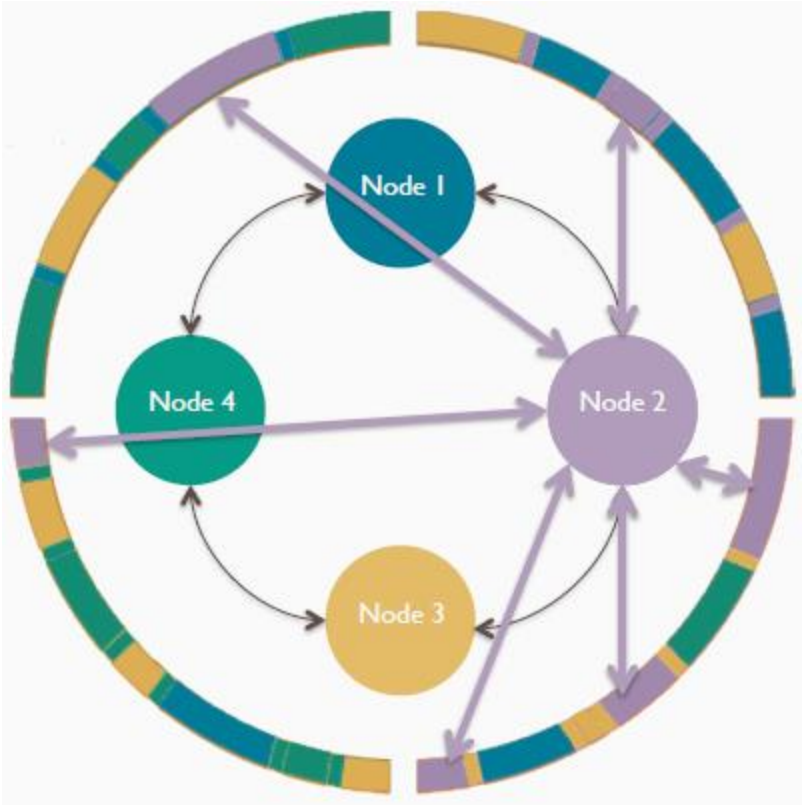
- Cassandra propose 3 partitionners
  - Murmur3Partionner (defaut): propose une distribution uniforme basé sur la fonction de hashage Murmur3
  - RandomPartionner: distribution uniforme basé sur un hashage MD5
  - ByteOrderedPartionner (historique) distribution lexical basé sur les octets des clés

- Murmur3Partionner est la fonction à utliser
- Le partitionner se configure dans le fichier cassandra.yaml
  - Il doit impérativement être le même dans tous les nœuds du cluster
  - partitionner: org.apache.cassandra.dht.Murmur3Partitioner

Que sont les noeuds virtuels? (1)

- De multiple petits segments primaires d'intervalle de token répartis sur tous les nœuds à la place d'un très grand intervalle
  - Les nœuds virtuels se comportent comme des nœuds classiques
  - La valeur par défaud est 256 par nœud

- En quoi les nœuds virtuels sont utiles?
  - Accélère le bootstrap des nœuds
  - L'impact d'une perte d'un nœud virtuel est réparti sur l'ensemble du cluster
  - Permet l'automatisation de l'affectation des plages de token



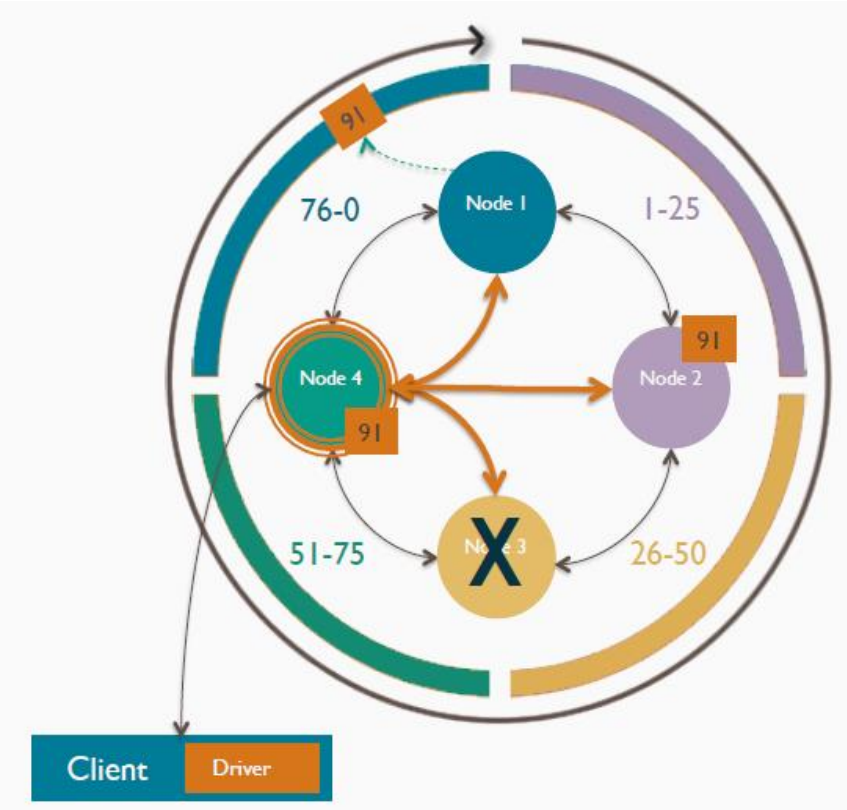
Que sont les noeuds virtuels? (2)

- Les nœuds virtuels sont activés dans le fichier cassandra.yaml
  - Les partitions, les nœuds classiques et les nœuds virtuels sont tous identifiés par un token
  - Les nœuds classiques ou virtuels par la plus haute valeur de son intervalle primaire
  - Le paramètre est :
    - num_tokens: 256

Indisponibilité d’un nœud

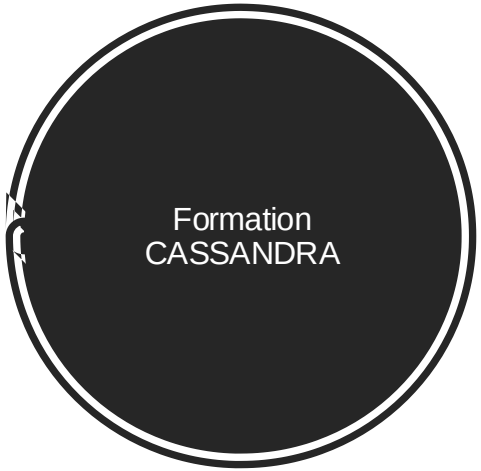
Qu’est ce qu’un hinted handoff? (1)

- Un mécanisme de récupération d’écritures dont le nœud de destination est inaccessible
- Le coordinateur peut stocker un ‘hinted handoff’
  - Si l’état ‘KO’ du nœud cible est connu
  - Ou s’il échoue à acquitter l’écriture
- Le coordinateur stock les hints dans sa table system.hints
- Les écritures sont rejouées lorsque le nœud redevient joignable.



Qu’est ce qu’un hinted handoff? (2)

- Un hinted handoff se compose:
  - Le nœud initialement cible qui est tombé
  - La partition initialement cible
  - Les données à écrire
- Ils se configurent dans le fichier cassandra.yaml:
  - hinted_handoff_enabled (defaut à true) : activés ou désactives par DC
  - max_hint_window_in_ms (defaut 3 heures soit 10 800 000 ms):
    - Passé ce délai de 3h, sans retour du nœud, il n’y a plus de HH enregistrés.
    - Et le node devra être réparé (repair) ou restorer pour redevenir consistant.

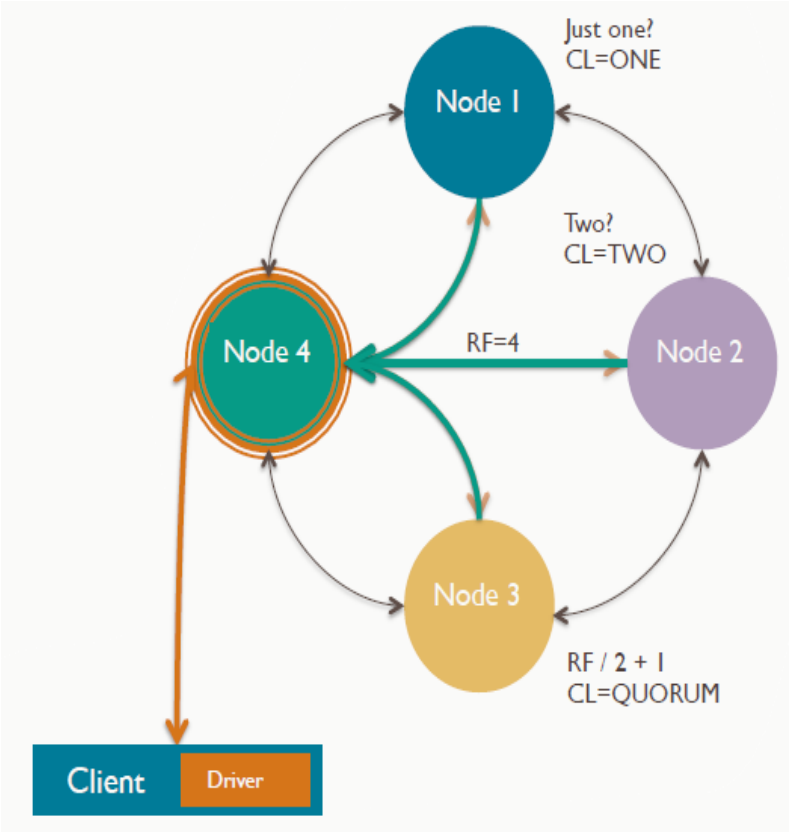


- Introduction - Rappel
- Cassandra architecture…
- La réplication des données
- La cohérence des données
- Chemins de lecture & écriture
- Mécanique de maintenace
- Cassandra & Spark

Consistance des données

Comprendre et tuner la consistance:

- Qu'est ce que la consistance?
  - La clé de partition détermine à quel nœud envoyé une requête
    - Le niveau de consistance (Consistency Level) : détermine combien de nœuds, contenant cette clé de partition, doivent acquitter son exécution pour indiquer au client sa bonne exécution (ou pas ☺) .
- L'acquittement dépend du type de requête:
  - En écriture : combien de nœuds doivent indiquer avoir reçu et écrit la requête
  - En lecture : combien de nœuds doivent acquitter en retournant les copies de données les plus récentes



Quels sont les principaux niveaux de consistance possibles?

Nom	Description	utilisation
ANY (écriture seulement)	Écrit sur n'importe quel nœud et si tous KO alors HH	Haute disponibilité Faible consistance en écriture
ALL	Écrire obligatoirement sur tous les nœuds sinon KO	Consistance élevée Faible disponibilité
ONE (TWO, THREE)	Vérifie le nœud le plus proche du coordinateur	Haute disponibilité Faible consistance en lecture
QUORUM	Vérifie l'atteinte du quorum avec les nœuds disponibles	Equilibre entre la consistance et la disponibilité
LOCAL_ONE	Vérifie le nœud le plus proche du coordinateur sur le DC local	Haute dispo et consistance sans traffic inter-DC
LOCAL_QUORUM	Vérifie l'atteinte du quorum avec les nœuds disponibles sur le DC local	Equilibre entre la consistance et la disponibilité sans traffic inter DC
EACH_QUORUM (écriture seulement)	Vérifie l'atteinte du quorum avec les nœuds disponibles sur tous le DC	Equilibre entre la consistance et la disponibilité avec du traffic inter DC

Comment positionne t-on le niveau de consistance par requête?

- Le niveau de consistance par requête est 1:
  - En cqlsh: la commande ‘CONSISTENCY’ avec les arguments ‘ONE’, ’QUORUM’, ‘ANY’, ‘ALL’, etc.. change le niveau de consistance pour toutes les requêtes durant la session cqlsh.
  - Avec un ‘driver’ client, on peut passer la variable ‘ConsistencyLevel’ à chaque requête

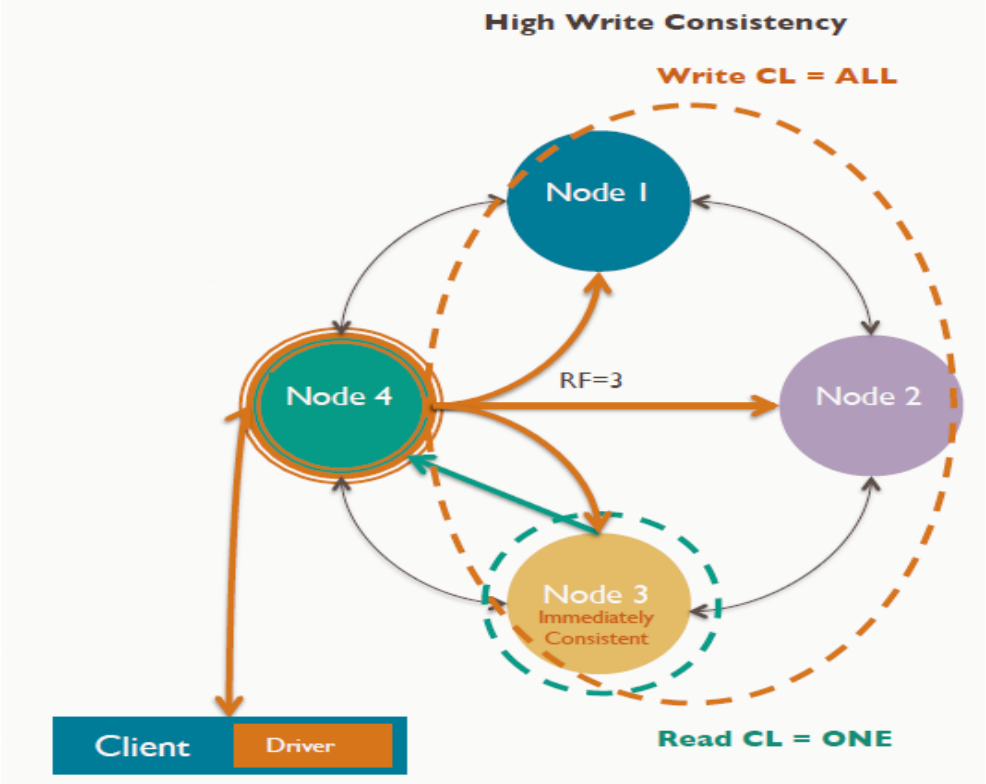
```
Connected to Test Cluster 3112 at dvihcdbspb00000.nor.fr.gin.intraorange:9042.
[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cassandra@cqlsh> use rotation_hot_histo_arch ;
cassandra@cqlsh:rotation_hot_histo_arch> consistency
Current consistency level is ONE.
cassandra@cqlsh:rotation_hot_histo_arch> consistency one;
Consistency level set to ONE.
cassandra@cqlsh:rotation_hot_histo_arch> consistency quorum;
Consistency level set to QUORUM.
cassandra@cqlsh:rotation_hot_histo_arch> consistency all;
Consistency level set to ALL.
cassandra@cqlsh:rotation_hot_histo_arch>
```

Qu’est ce que la consistance immédiate par rapport à la consistance à terme (eventually consistency en anglais)?

- Pour une lecture donnée, quelle est la probabilité que les données soient périmées?
- Consistance immédiate: les lectures retournent toujours les données les plus récentes
  - CL=ALL : garantie le niveau de consistance immédiate car tous les réplicas des nœuds sont accédés et comparés afin de ne renvoyer que les données les plus ‘fraiches’
  - Mais de ce fait offre le temps de latence le plus élevé
- Finalement consistant: les lecture peuvent, à un instant t, retourner des données périmées
  - CL=ONE: présente le niveau de risque le plus élevé, car seulement un réplicas est accédé
  - Mais temps de réponse excellent car très faible latence avec un seul nœud consulté

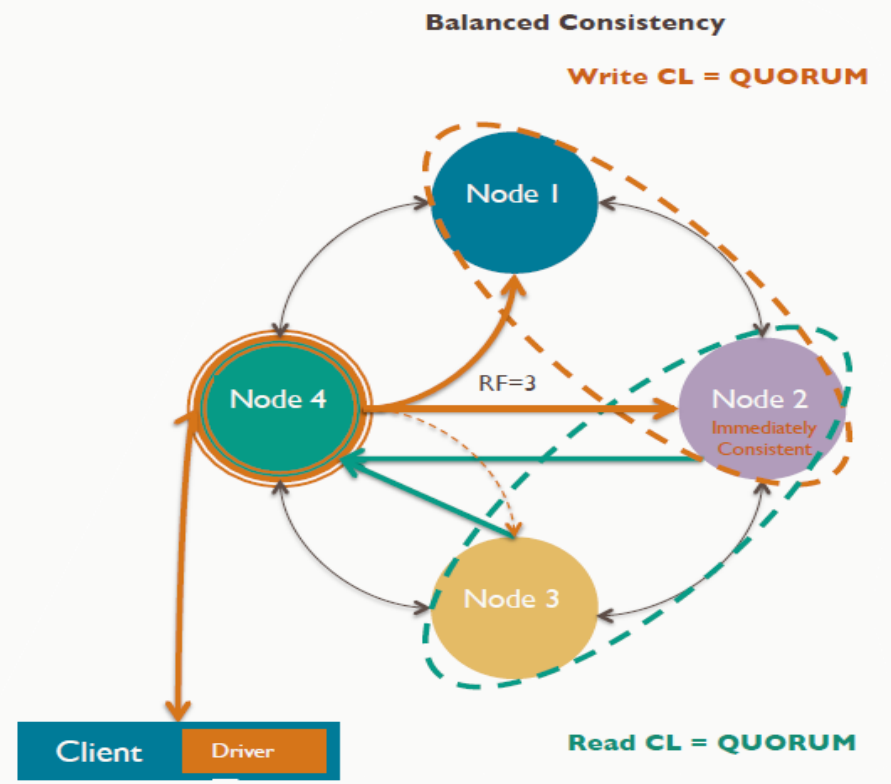
Que signifie régler la consistance? (1)

- Les lectures et les écritures doivent être positionnées à des niveaux de consistance spécifiques
- Formule de consistance :
  - Si (Nœuds_écrits + nœuds_lus) > RF alors la consistance est immédiate



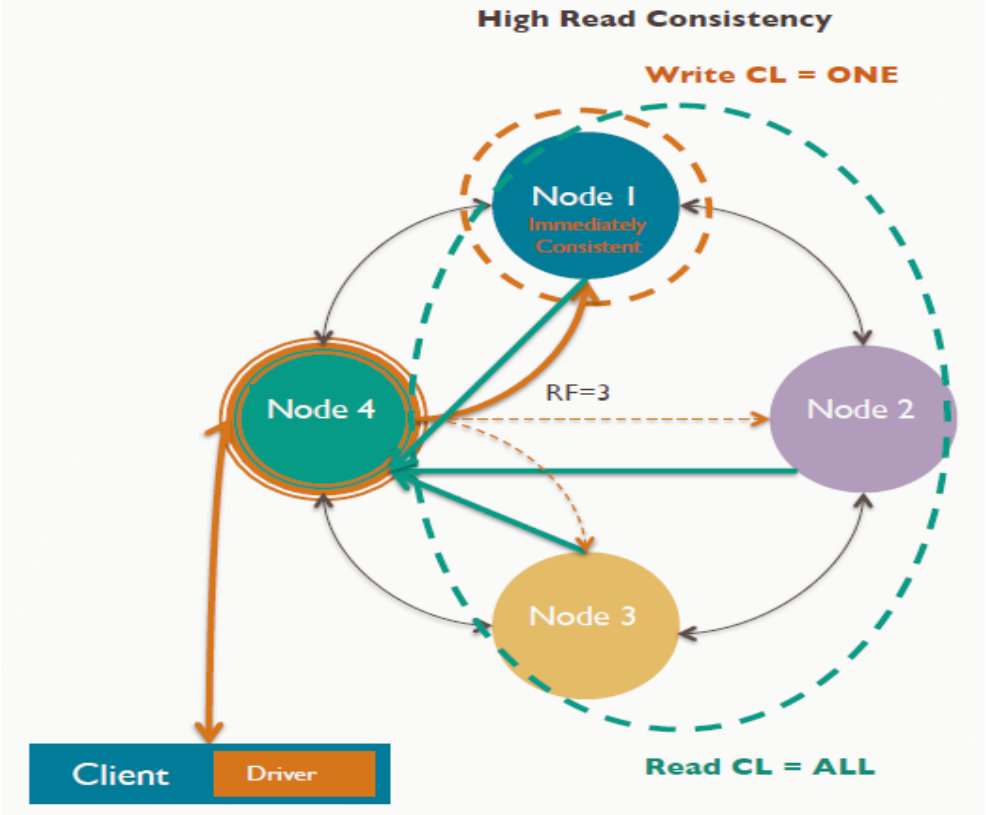
Que signifie régler la consistance? (2)

- Les lectures et les écritures doivent être positionnées à des niveaux de consistance spécifiques
- Formule de consistance :
  - Si (Nœuds_écrits + nœuds_lus) > RF alors la consistance est immédiate



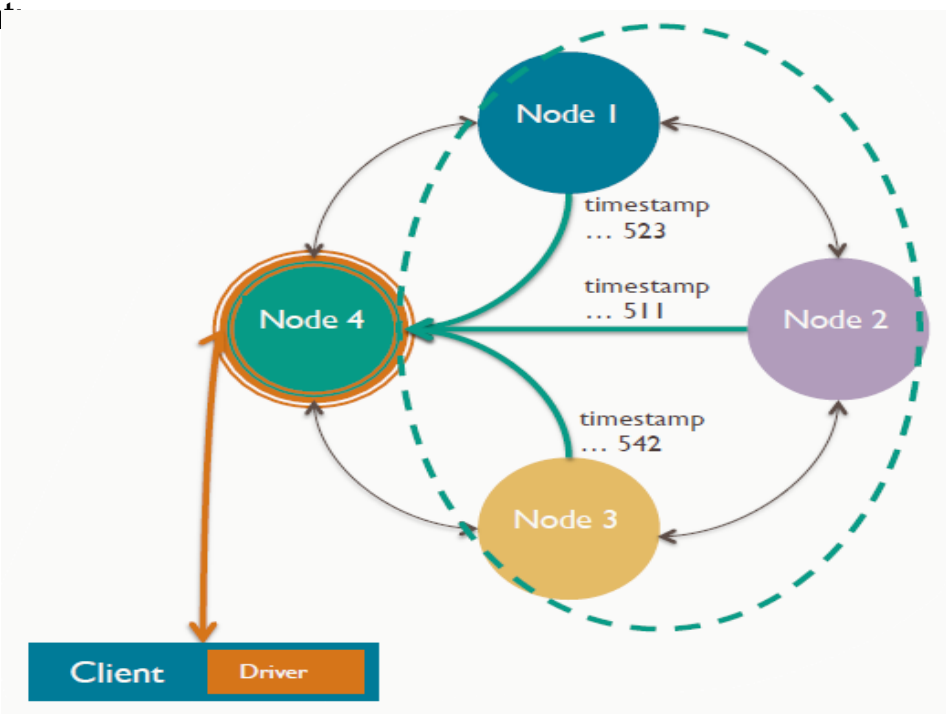
Que signifie régler la consistance? (3)

- Les lectures et les écritures doivent être positionnées à des niveaux de consistance spécifiques
- Formule de consistance :
  - Si (Nœuds_écrits + nœuds_lus) > RF alors la consistance est immédiate



Pourquoi la synchronisation des horloges des nœuds avec le serveur d'horloge est si importante?

- La synchronisation des horloges des nœuds C* est critique car:
  - Chaque écriture des colonnes d'une table contient:
    - Nom de la colonne
    - Valeur de la colonne
    - Horadatage (timestamp) depuis epoch (01/01/1970)
    - L'enregistrement le plus récent est retourné au client

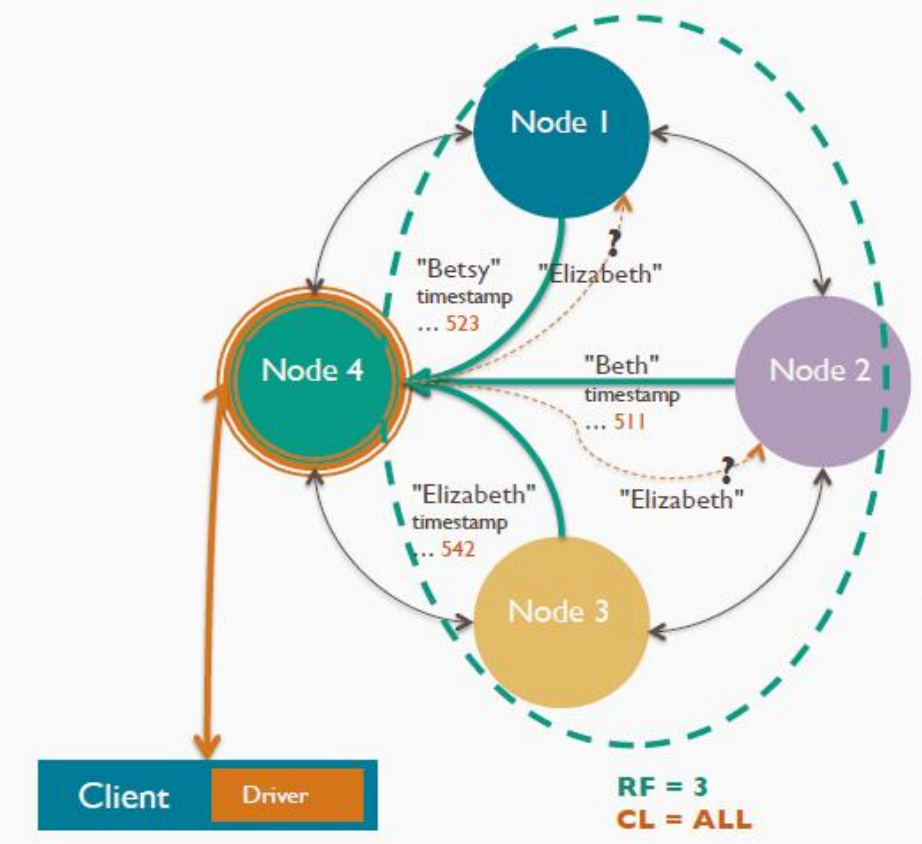


Comment choisir un niveau de consistance?

CL ONE	CL QUORUM	CL ALL
Faible latence	Plus haute latence que ONE	Latence la plus forte
Débit élevé	Débit faible	Débit le plus faible
Disponibilité la plus forte	Disponibilité plus élevé que ALL	Disponibilité la plus faible
Risque de lectures périmées si : (Read_CL + Write_CL < RF)	Pas de lecture périmées si: (READ_CL et WRITE_CL sont à QUORUM)	Pas de lecture périmées si : (soit READ_CL ou WRITE_CL sont à ALL)

Introduction aux opération dites « d'anti-entropie »

- Qu'est qu'un read-repair?
- Lors d'une lecture, un 'digest query' est envoyé aux réplicas afin qu'ils comparent avec le leur et se mettent à jour si nécessaire:
  - 'Digest query ': correspond à une valeur des données hachée plutôt que d'envoyer toutes les données



Comment lancer un ‘repair’ avec la commande notetool? (1)

- La commande ‘nodetool repair’ rend consistant les données d’un nœud avec les données les plus récentes des autres réplicas dans le cluster
- Les clusters avec un taux élevé d’écritures et de suppression avec un niveau de consistance < ALL nécessitent des ‘repairs’ réguliers
  - nodetool –h [host] –p [port] repair [options]

- Les options permettent d’alléger la charge disque liée au ‘repair’
  - --partionner-range : restreint l’opération aux seuls segments de token primaire
  - --start-token [uuid] –end-token [uuid] : limite le repair à cet intervalle de token
  - -dc [name] ou [local] : pour cibler la ‘réparation’ à un DC précis ou bien le local

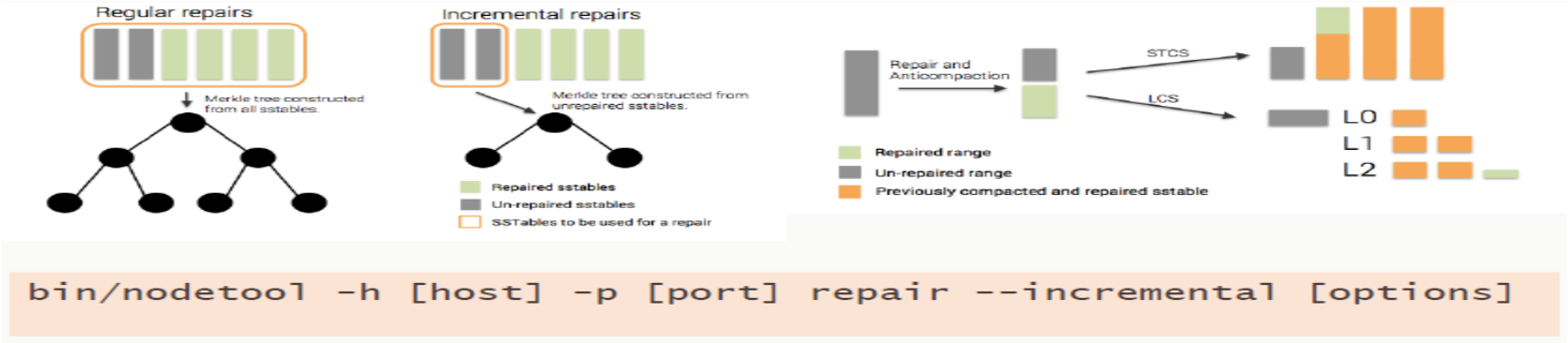
Comment lancer un ‘repair’ avec la commande notetool? (2)

- Quand lancer ‘nodetool repair’ ?
  - Réparer un nœud défaillant
  - Ré-inclure un nœud tombé dans le cluster
  - Périodiquement sur les nœuds avec des lectures peu fréquentes
  - Périodiquement sur des nœuds avec écritures et suppressions

- Si l’exécution est périodique alors il faut le faire au moins à chaque délai ‘gc_grace_seconds’:
  - Gc_grace_seconds : purge des tombstones tous les 10 jours par défaut (864 000sec)
  - Tombstones : marqueur de suppression sur une colonne supprimée dans une partition
  - Ne pas ‘réparer’ pendant le délai de grâce des ‘tombstones’ peut conduire à la résurrection de lignes fantômes!
  - Car l’ordre de suppression peut ne pas avoir été propagé à tous les nœuds, avant que les tombstones soient supprimés lors de la compaction
  - Ex : ALTER TABLE performer WITH gc_grace_seconds = 432000 ; (5 jours)

Qu’est qu’un repair incrémental? (1)

- Répare les données qui non pas déjà été réparées
  - Merkle Tree sont seulement générés et comparés que pour les SSTables non réparées
  - Les Sstables non réparées sont identifiées par la propriété ‘repairedAt’ des métadonnées
  - Une anti-compaction se produit après le ‘repair incrémental’ afin de séparer les intervalles de token réparés des intervalles non réparés

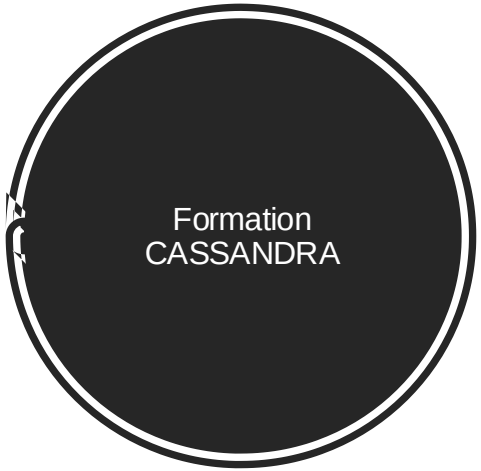



```
bin/nodetool -h [host] -p [port] repair --incremental [options]
```

Qu’est qu’un repair incrémental? (2)

- Les outils pour les Sstables lorsqu’on utilise ‘incrémental repair’:
  - sstablemetadata : affiche le ‘repairedAt’ d’une sstable
  - sstablerepairedSet : pour modifier manuellement le status d’une Sstable

- Guide pour utiliser l’incrémetal repair :
  - Recommandé avec la compaction Leveled Compaction
  - À lancer quotidiennement
  - Eviter ‘l’anti-compaction’ en n’utilisant pas les options de ‘partionner range’ ou ‘subrange’



- Introduction - Rappel
- Cassandra architecture…
- La réplication des données
- La cohérence des données
- **Chemins de lecture & écriture**
- Mécanique de maintenace
- Cassandra & Spark

INSTALLATION CASSANDRA

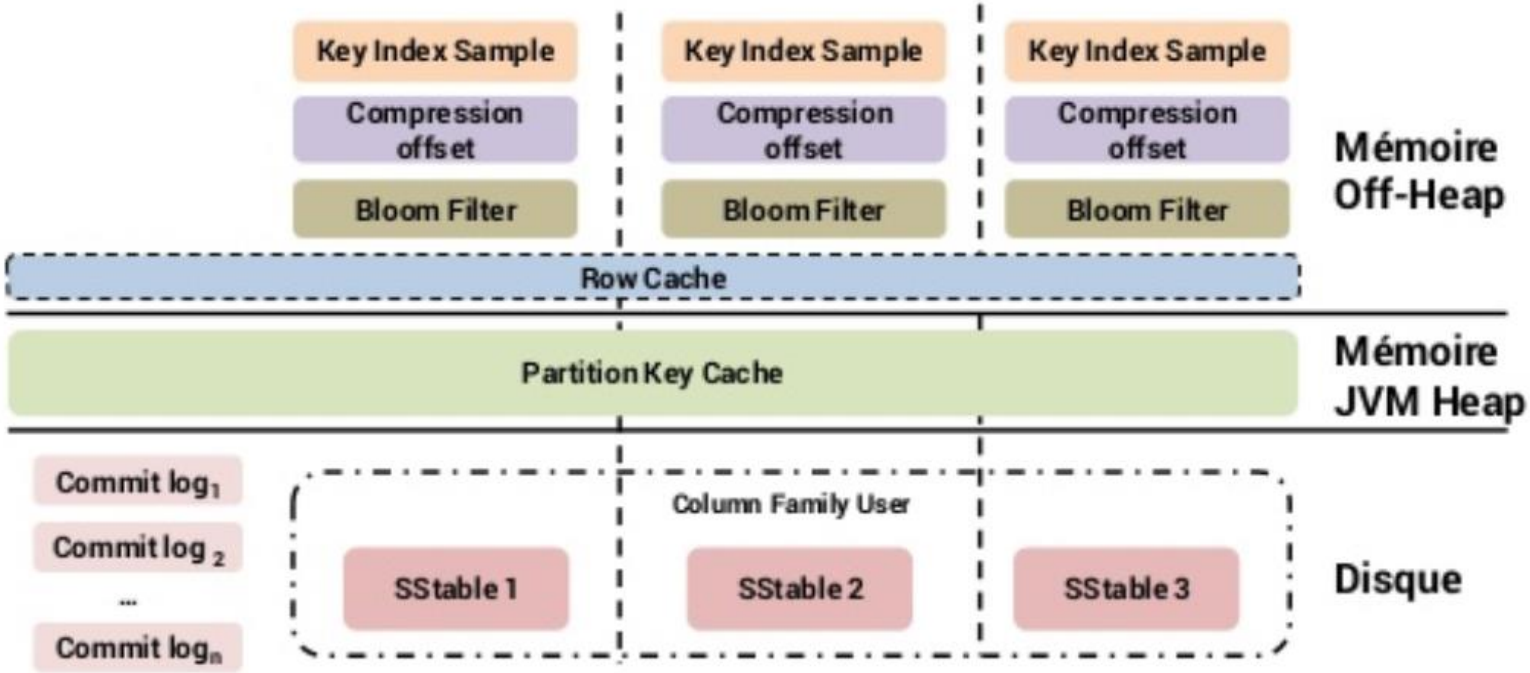
Cloud : ASTRADB

DOCKER

PHYSIQUE

Chemin de lecture & Ecriture

Format de fichiers

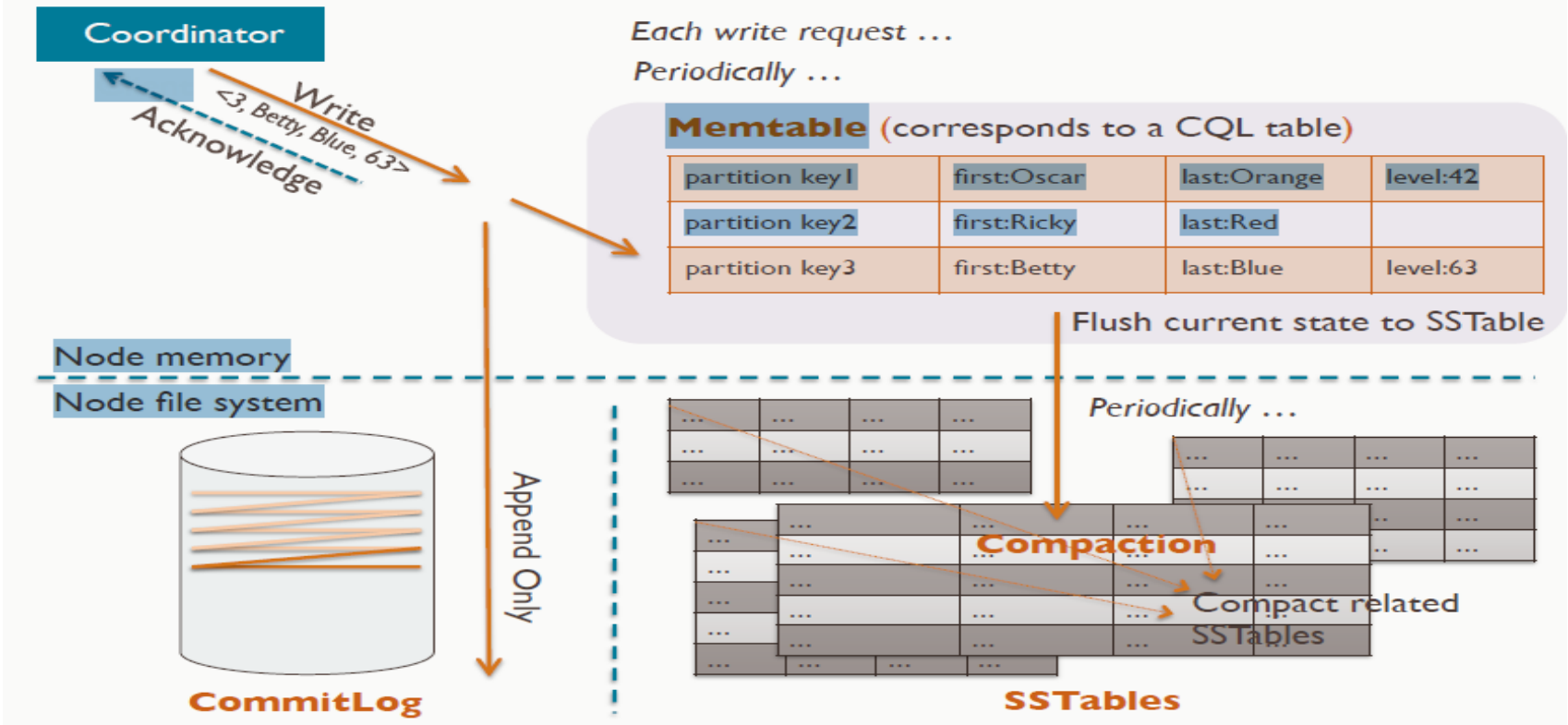


Quels sont les éléments clés du processus d'écriture?

- Chaque nœud implémente 4 composants clés pour gérer ses écritures:
  - Memtables: tables en mémoire correspondantes au table du CQL avec les indexes éventuels
  - CommitLog: une log en mode ajout, servant à la réintroduction d'un nœud défaillant
  - SSTables:c'est un clichés des memtables fait périodiquement sur disque, libérant la 'HEAP' de la JVM
  - Compaction: c'est un mécanisme de fusion et de de réduction des SSTables

- Quand un nœud reçoit une requête en écriture:
  - 1 : l'enregistrement est ajouté dans le commitlog et
  - 1': l'enregistrement est inséré dans la memtable associé à sa table
  - 2: régulièrement les memtables sont écrites sur disques en fichiers Sstables; libérant la HEAP et les segments associés au commitlog
  - 3: régulièrement le mécanisme de compaction fusionne les Sstables.

Vue schématique des écritures sur un nœud:



Format de fichiers

Format

- <keyspace>-<column family>-<version>-<génération>-<composant>.db

Méta-données

test-user-ic-1-Filter.db → filtre de Bloom
test-user-ic-1-Index.db → index des offset de **#partition** dans le SSTable
test-user-ic-1-Statistics.db → statistiques
test-user-ic-1-Summary.db → échantillon des offsets de **#partition**
test-user-ic-1-TOC.txt

Qu’est qu’un commitlog et comment est il configuré?

- C’est un fichier en mode ‘append’ utilisé pour reconstruire automatiquement les ‘memtables’ lors du redémarrage d’un nœud tombé.

- Il est configuré dans le fichier cassandra.yaml

- Les ‘memtables’ sont flushés sur disque quand la taille max du ‘CommitLog’ est atteinte :
  - Commit_log_total_space_in_mb:
  - Commitlog_segment_size_in_mb: taille max d’un segment de commitlog (defaut : 32 Mb)

- Les entrées dans les segments sont marquées comme ‘flushées’ dès qu’une memtable devient une SSTable
  - Les segments mémoires sont régulièrement recyclés

- Séparer les fs de ‘commitlog’ sur des disques physiquement distincts de celui des ‘data’ est important. Sauf si vous utilisez des SSD.
  - Commit_log_directory : défini l’emplacement des fichiers commitlog.

Qu’est ce qu’une SSTable et quelles sont ses caractéristiques (1)

- Un SSTable (« Sorted String Table ») est :
  - Un fichier trié de partitions qui est ‘immutable’
  - Ecrit sur disque par des écritures séquentielles très rapides
  - Contient le contenu d’une Memtable au moment du ‘flush’
- Le contenu d’une table CQL comprends donc :
  - Le contenu de sa Memtable en mémoire
  - Plus, toutes les Sstables flushées depusi sa Memtable
- Les sstables sont périodiquement compactées depuis plusieurs vers une ou quelques unes.

Qu’est que la compaction?

- Les updates modifie la Memtable associée à un table CQL
  - pas les fichiers SSTables déjà générés lors du flush sur disque du contenu de cette memtable
  - Les mises à jours seront flushées dans une nouvelle SSTable
- Donc les Sstables doivent être régulièrement compactées:
  - Les x Sstables générées seront fusionnées
  - Les colonnes avec la version la plus récente seront traitées dans une partition d’une nouvelle SSTable
  - Les partitions marquées pour suppression (« Tombstone ») seront effectivement supprimées
  - Les anciennes SSTables, devenues inutiles, seront supprimées.

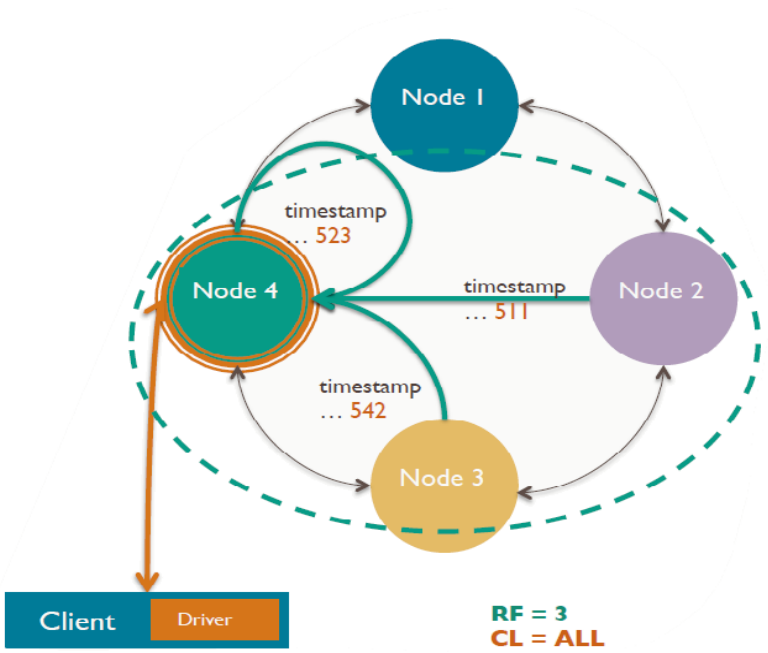
  - Nous essayerons de revoir les différents types de compaction plus tard dans cette formation.

La mécanique des lectures avec Cassandra

Lire des données depuis Cassandra:

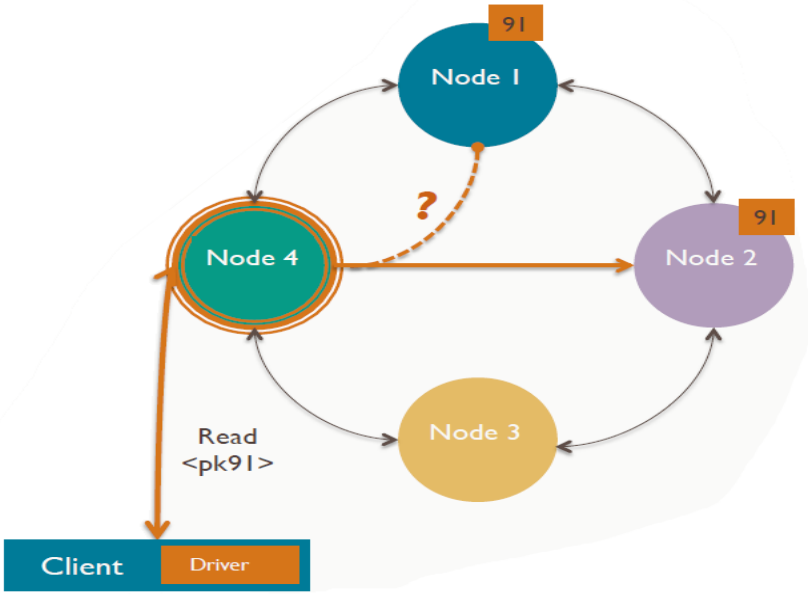
- Cassandra renvoi l'enregistrement le plus récent parmi les nodes lus pour une requêtes donnée.

- En tenant compte du niveau de consistance demandé, le nombre de node à lire sera fixé.
- Le niveau de consistance peut être variable pour chaque requête



La mécanique des lectures avec Cassandra (2)

Si un nœud est trop lent à retourner une réponse, alors le coordinateur transfert la requête vers un autre nœud contenant un répliquât de la partition requêtée (seulement si RF > 1).



Quels sont les éléments clés du cheminement de lecture C* ?

Chaque nœud implémente, pour chaque Table, les structures mémoires suivantes :

- Memtable : contient les données les plus récentes et fait partie du processus de merge*
- Rowcache: contient les données des lectures les plus récentes (optionnel)
- Bloom Filters: qui indique si une clé de partition peut être présente dans sa SSTable
- Key Caches: établi les liens entre les clé de partiton les plus récemment lues et leurs positions dans leur fichiers SSTables
- Partition Summaries: contient un échantillonnage du ‘partition index’

Chaque nœud implémente, pour chaque Table, les composants suivants sur disque:

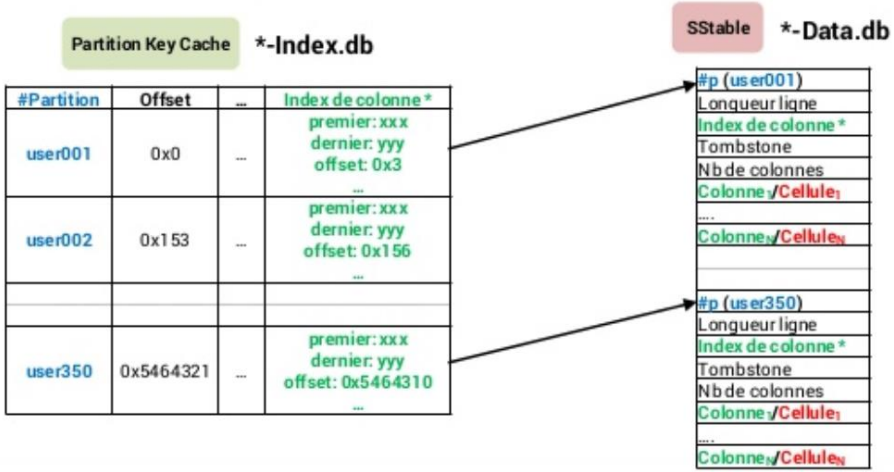
- Partition Indexes : une correspondance des clé de partition triée avec leur position dans la SSTable
- SSTables: qui est un fichier généré,périodiquement, à partir d’un flush de la memtable

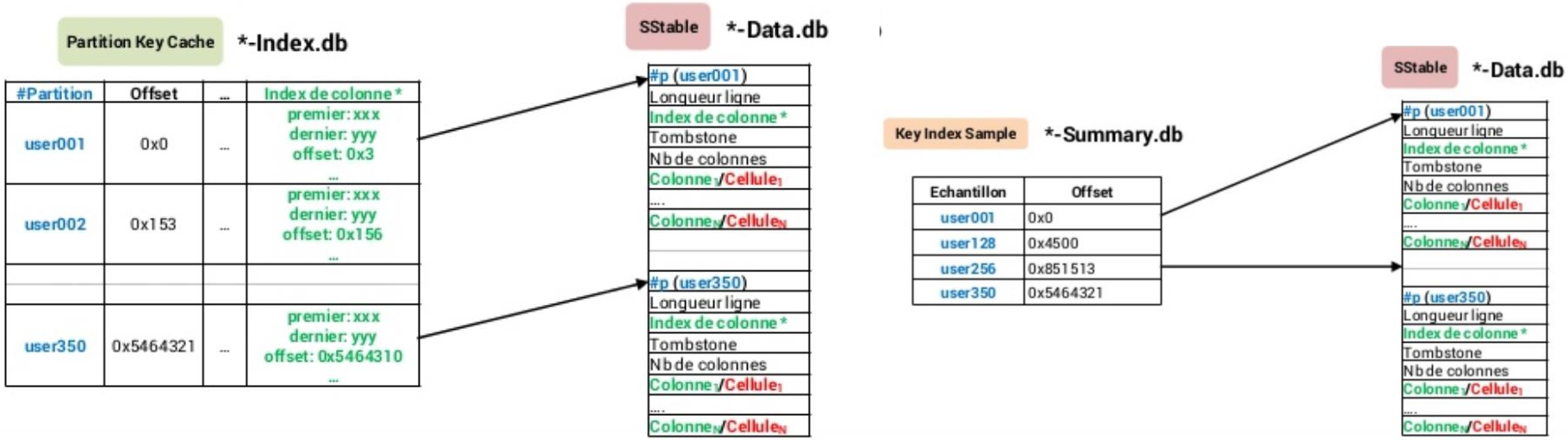
Merge:

- A moins d’être trouvée dans le ‘row cache’, une lecture cassandra utilisera la clé de partition pour localiser, ‘merger’ et renvoyer les valeurs depuis une memtable et chaque SSTable qui héberge cette valeur de clé.

Qu’est ce qu’une SSTable et quelles sont ses caractéristiques (2)

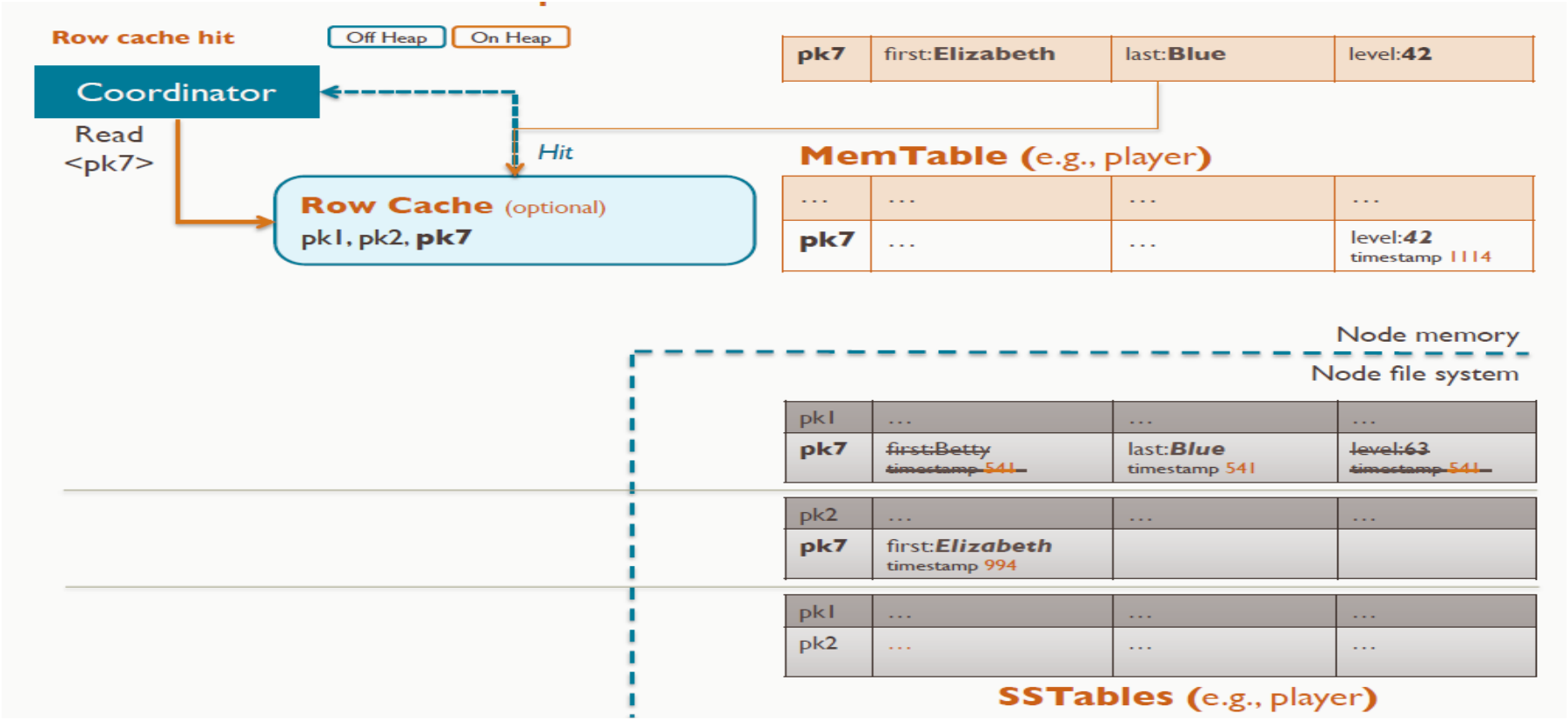
- Pour chaque Sstable, deux structures sont créées:
  - Partition index :
    - Qui contient : la liste de toutes ses clé primaires et les positions de départ de chaque lignes
  - Partition summary :
    - Contient un échantillonnage de ses partitions d’indexes (par défaut 128 clés primaires par partition)



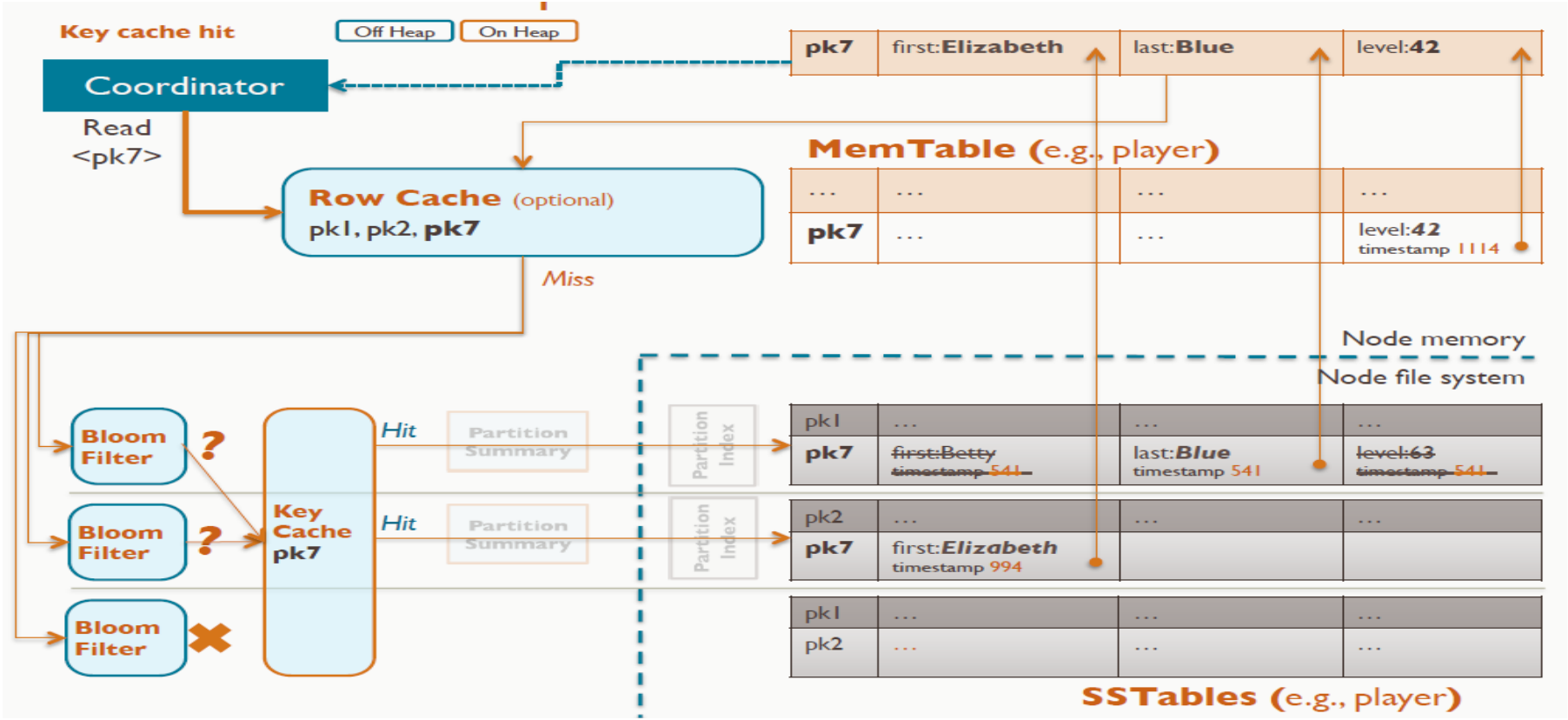


Memtable : contient les données les plus récentes et fait partie du processus de merge*
Rowcache: contient les données des lectures les plus récentes (optionnel)
Bloom Filters: qui indique si une clé de partition peut être présente dans sa SSTable
Key Caches: établi les liens entre les clé de partiton les plus récemment lues et leurs positions dans leur fichiers SSTables
Partition Summaries: contient un échantillonnage du 'partition index'

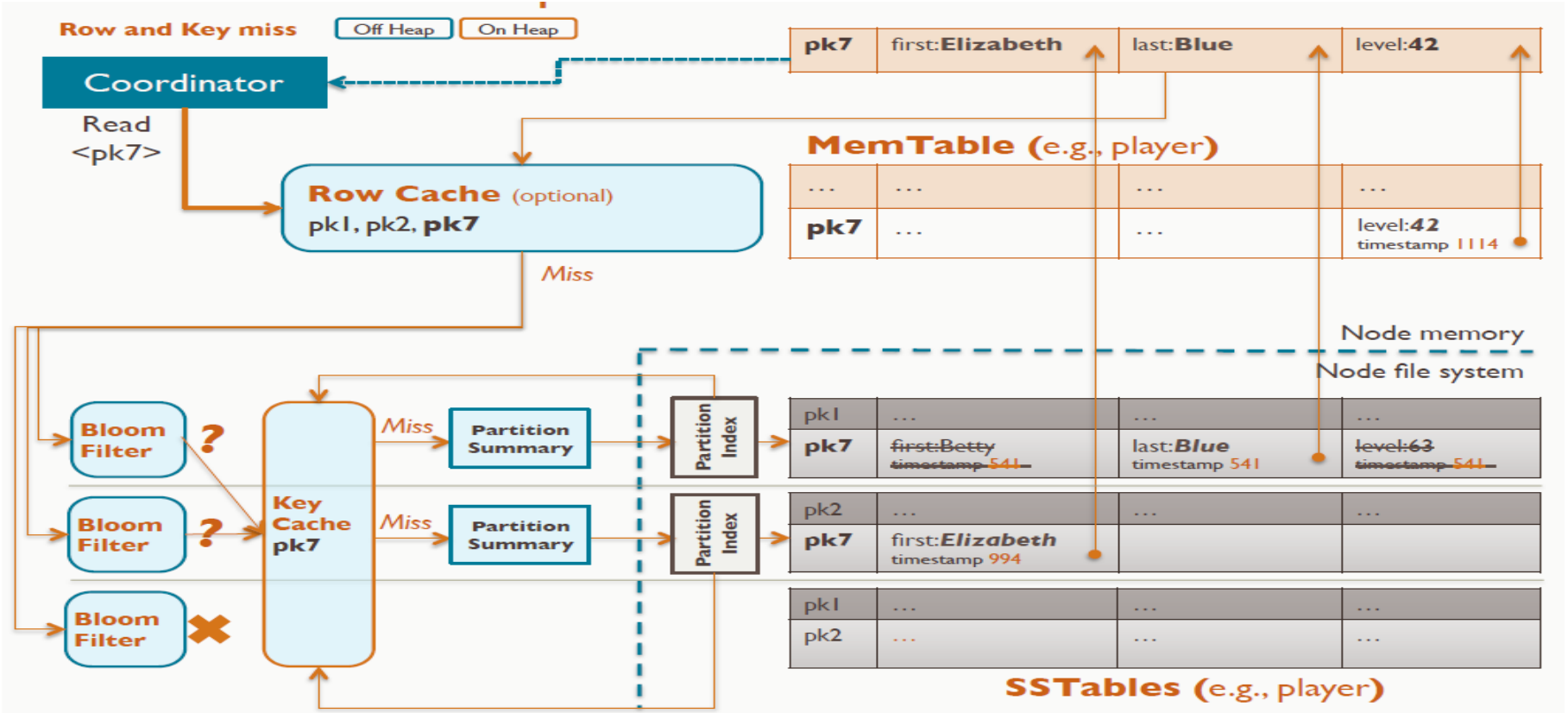
Le cheminement de lecture sur chaque nœud:



Le cheminement de lecture sur chaque nœud: (2)



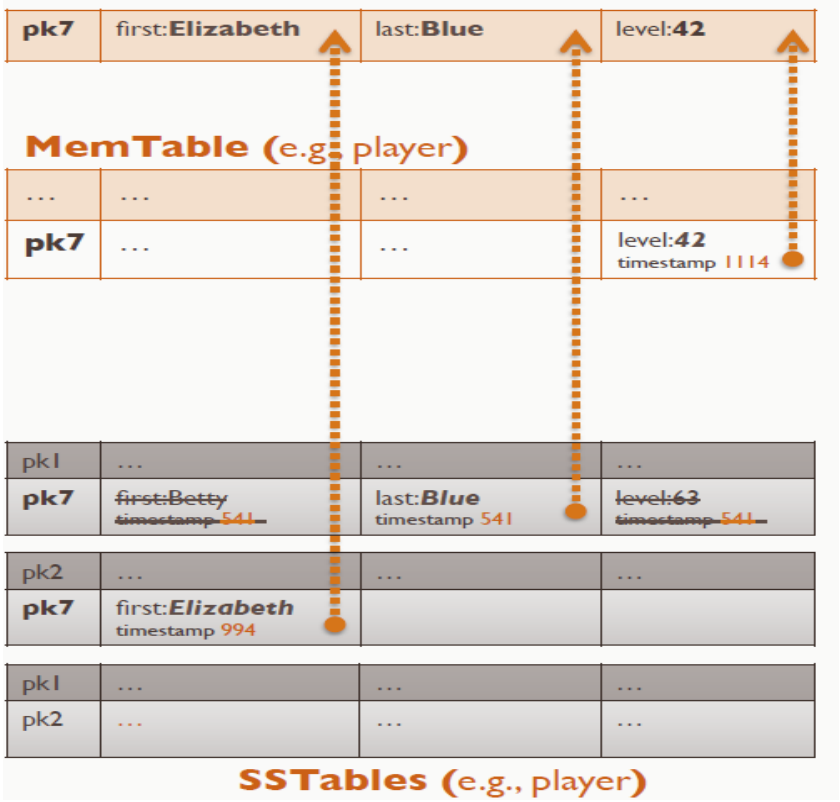
Le cheminement de lecture sur chaque nœud: (3)



Comment une Memtable et ses SSTables sont utilisées pendant une lecture?

La memtable et ses Sstables sont parcourues lors de la lecture d'une clé de partition:

-les valeurs des colonnes les plus récentes sont recombinées pour fournir le résultat



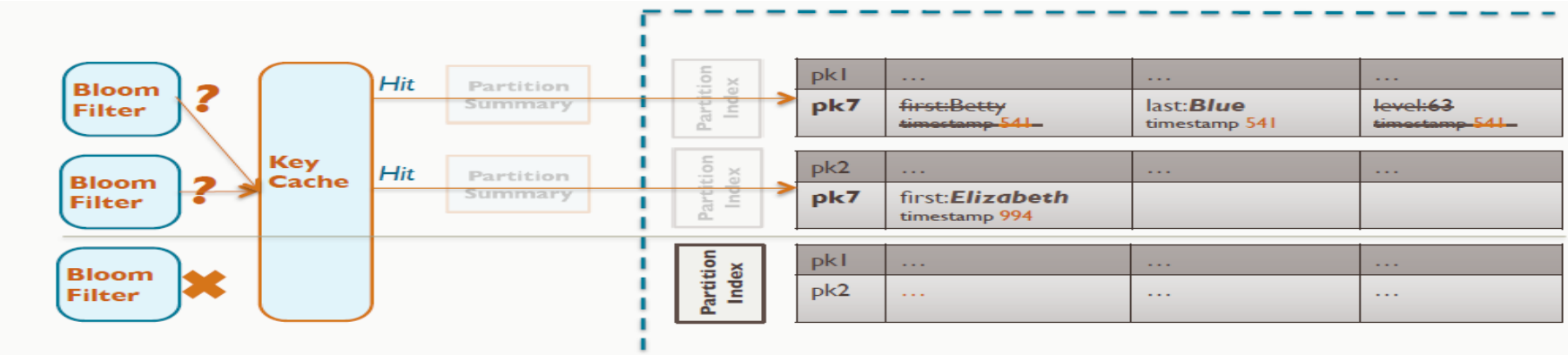
Chemin de lecture

Qu’est ce qu’un Bloom Filter et comment optimise t-il une lecture?

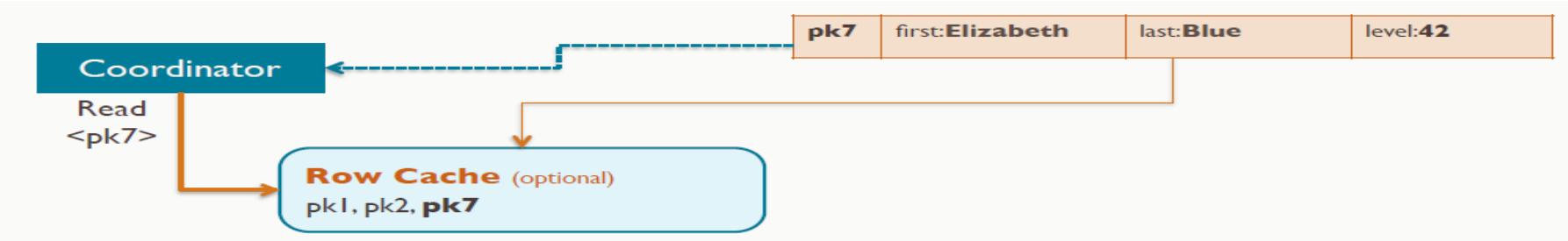
C’est une structure servant à déterminer la probabilité de présence d’un clé de partition dans une Sstable:

- Chaque SSTable a un fichier ‘Bloom Filter’ sur disque qui sera monté en mémoire ‘off-heap’ de la JVM C*

Ils éliminent ainsi le parcours de certains des fichiers SSTables à la recherche d’une clé de partition:



Qu’est ce que le ‘Row Cache’ et comment est il configuré?



- Les lignes fusionnées (mergées) d’une clé de partition sont sauvegardées dans la zone ‘off-heap’ de la JVM.
- La mise en cache des lignes est activée avec les propriétés ‘caching’ et ‘row_per_partition’ de la table:
  - ALL : toutes les lignes pour une clé de partition donnée
  - n: les n premières lignes pour clé de partition
  - NONE: (valeur) par défaut. Cache des lignes désactivé.

```
CREATE TABLE player (
   first text PRIMARY KEY,
   last text,
   level text
)
WITH caching = {'keys': 'ALL',
'rows_per_partition': '1'};
```

Qu’est ce que le ‘Row Cache’ et comment est il configuré? (2)

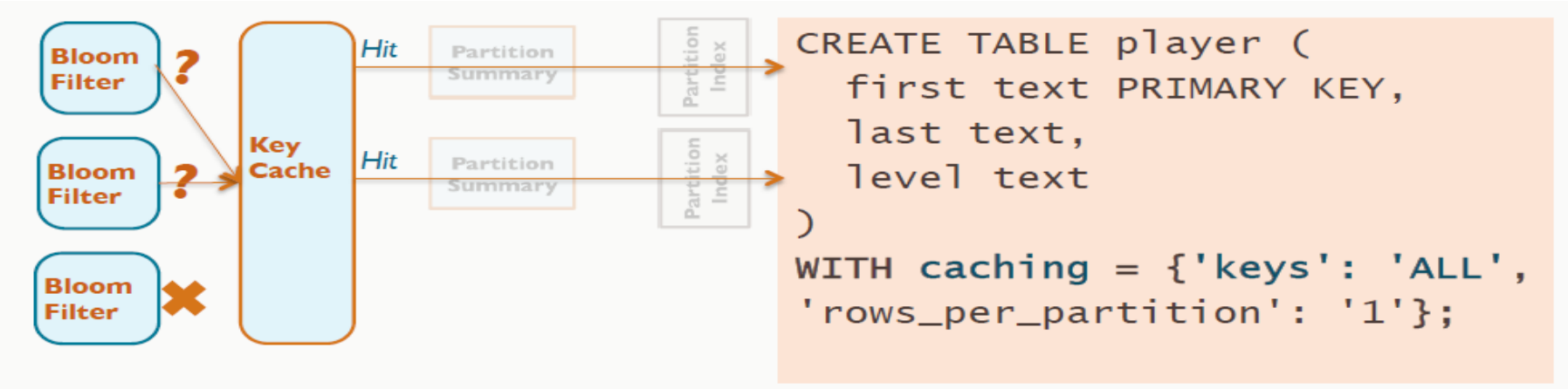
Les caches peuvent être régulièrement sauvegardés sur disques pour réduire le temps de redémarrage des nœuds.

La taille du cache et la fréquence de sauvegarde sont positionnés pour toutes les tables d’un nœud dans le fichier cassandra.yaml:

- row_cache_size_in_mb : taille maximal du cache. 0 pour le désactiver.
- row_cache_save_period : périodicité de backup du cache, exprimé en sec, dans le répertoire saved_cache_directory. 0 pour désactiver.
- row_cache_keys_to_save : nombre maximum de lignes à sauvegarder à chaque période. Si 0 alors toutes les lignes seront sauvegardés.
- saved_cache_directory: endroit où seront backupés les lignes, clés et counteur.

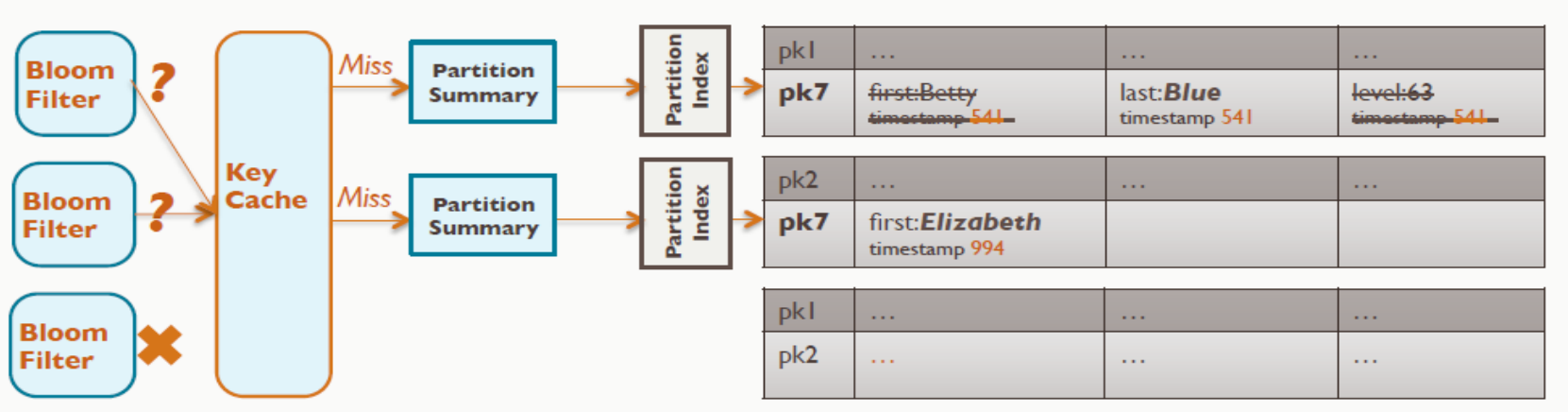
Qu’est ce que le ‘Key Cache’ et comment est il configuré?

- Le ‘key cache’ sauvegarde une clé de partition et ses positions dans les différentes SSTables pour une Memtable:
  - Une entrée dans le ‘key cache’ pour chaque SSTable contenant la clé de partition
  - Réduit les temps de lecture par un simple ‘seek’ par répliquas récemment accédé.
- L’activation du ‘key cache’ est faite avec les paramètres ‘caching’ et ‘keys’ de la table :
  - ALL: (défaut) active le ‘key cache pour cette table’
  - NONE: désactive le ‘key cache’



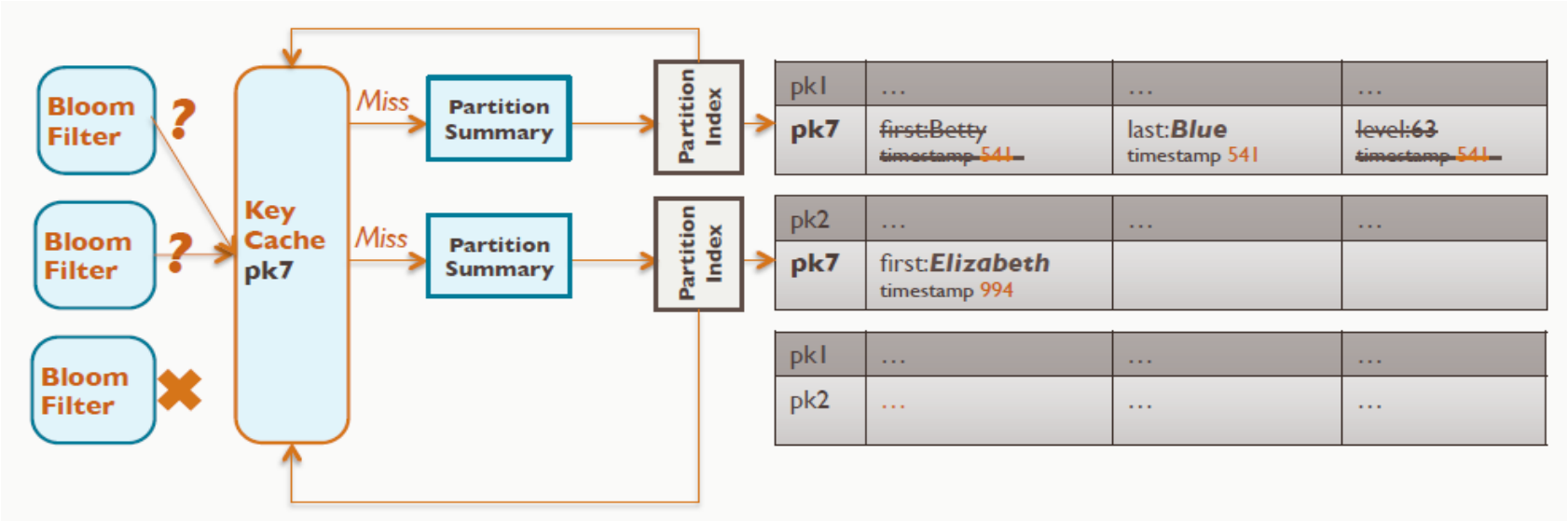
Qu’est ce que les ‘Partitions Summaries’ et comment sont elles utilisées?

- Si la position d’une clé de partition n’est pas dans le ‘key cache’ alors la lecture doit lancer la lecture sur disque (en anglais ‘seek’)
- La ‘Partition Summary’ est une structure montée en mémoire correspondante à un échantillonnage du ‘Partition Index’, utilisée pour approximer la position de la clé dans le fichier de ‘partition d’index’:
  - La valeur par défaut de l’échantillonnage : 1 pour 128 clé de partitions dans l’indexe.
  - Elle est configurée avec la propriétés min_index_interval (défaut: 128) et max_index_interval (default : 2048)
  - Cette structure est montée en mémoire de type ‘off-heap’



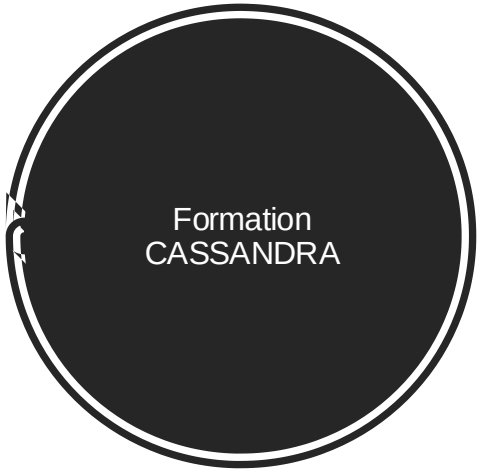
Qu’est ce que les ‘Partitions Index’ et comment sont elles utilisées?

- Le ‘partition index’ de chaque SSTables fourni la position physique de chaque clé de partitions, triée par clé de partition.
- En démarrant à partir de l’approximation de la position, lue dans la ‘Partition Summary’; le ‘partition index’ est parcouru pour trouver la position physique dans la SSTable de la clé de partition:
  - Une fois que la position est trouvée, elle est ajoutée dans le ‘Key Cache’.



Résumé:

- Une clé de partition trouvée dans le ‘partition index ’ est ajoutée au ‘Key Cache’
- C* fusionne les colonnes les + récentes entre une memtable et ses SSTables
- Si le ‘row Cache’ est actif, alors il sera mis à jour après la fusion des colonnes
- Les caches des lignes et des clés sont contrôlés par les paramètres de table.



- Introduction - Rappel
- Cassandra architecture…
- La réplication des données
- La cohérence des données
- Chemins de lecture & écriture
- Mécanique de maintenace
- Cassandra & Spark

Objectifs d’apprentissage:

- Comprendre les ‘tombstones’ et la suppression
- Comprendre la compaction et sa nécessité
- Choisir et implémenter une stratégie de compaction

Que sont les ‘tombstones’ et comment sont ils utilisés?

- Les colonnes supprimées ne le sont pas immédiatement, elles sont simplement marquées:
  - Car une suppression immédiate aurait été couteuse en accès disques
- Quand un ordre CQL requête pour une suppression de colonnes dans une partition, ou que leur TTL est expiré lors d’un requête en lecture alors:

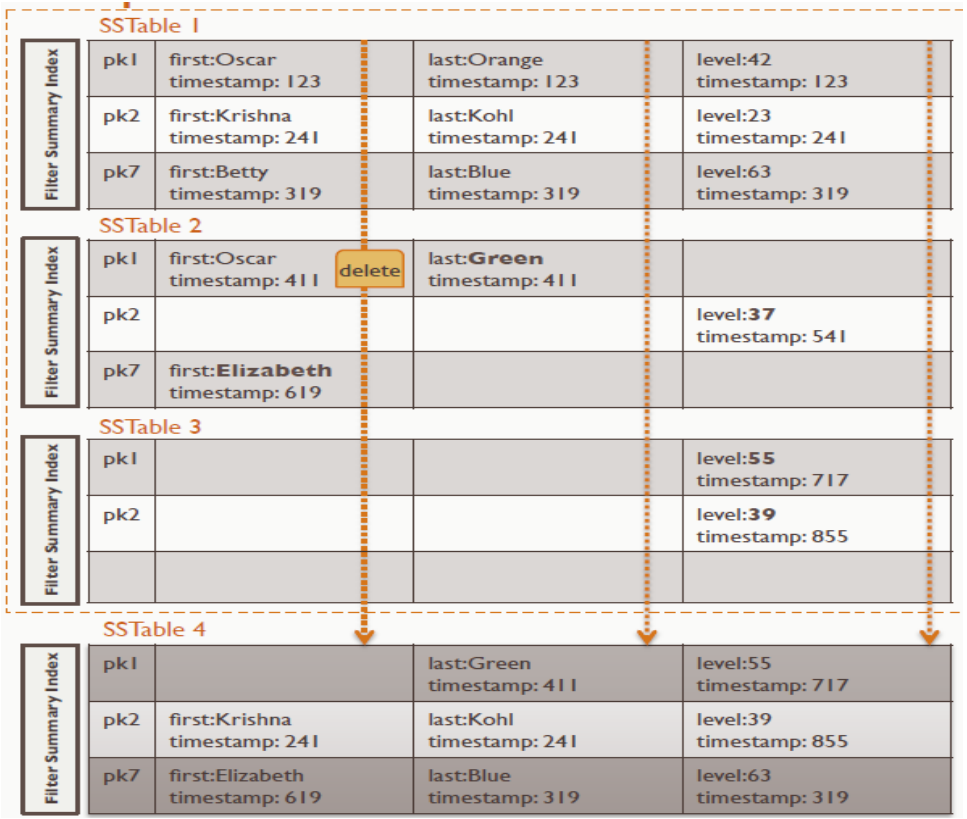
  1) Un marqueur appelé ‘tombstone’ est appliqué sur cette colonne en Memtable
  2) les requêtes suivantes considérerons cette ligne comme supprimée
  3) lors du prochain ‘flush’ de la memtable le tombstone est enregistré dans la SSTable
  4) à chaque compaction, les ‘tombstones’ plus agés que la valeur de ‘gc_grace_seconds’ sont évincés de la nouvelle SSTable résultante.

Que sont les ‘tombstones’ et comment sont ils utilisés? (2)

- La propriété ‘gc_grace_seconds’ de la table définie la durée de retention des tombstones avant leurs évictions à la prochaine compaction. La valeur par défaut est de 10 jours, soit 864 000 secondes)
  - ‘les colonnes zombies’ sont des colonnes supprimées qui réapparaissent. Comment?
    - Si un nœud tombe avant qu’il ne reçoive le ‘tombstone’ de suppression qu’il aurait dû recevoir, et qu’il soit redémarré après la période de ‘gc_grace_seconds’ alors il va restorer les lignes que les autres nœuds auront déjà purgés.
    - Solution : lance un nodetool repair lorsque l’on restore un nœud défaillant.

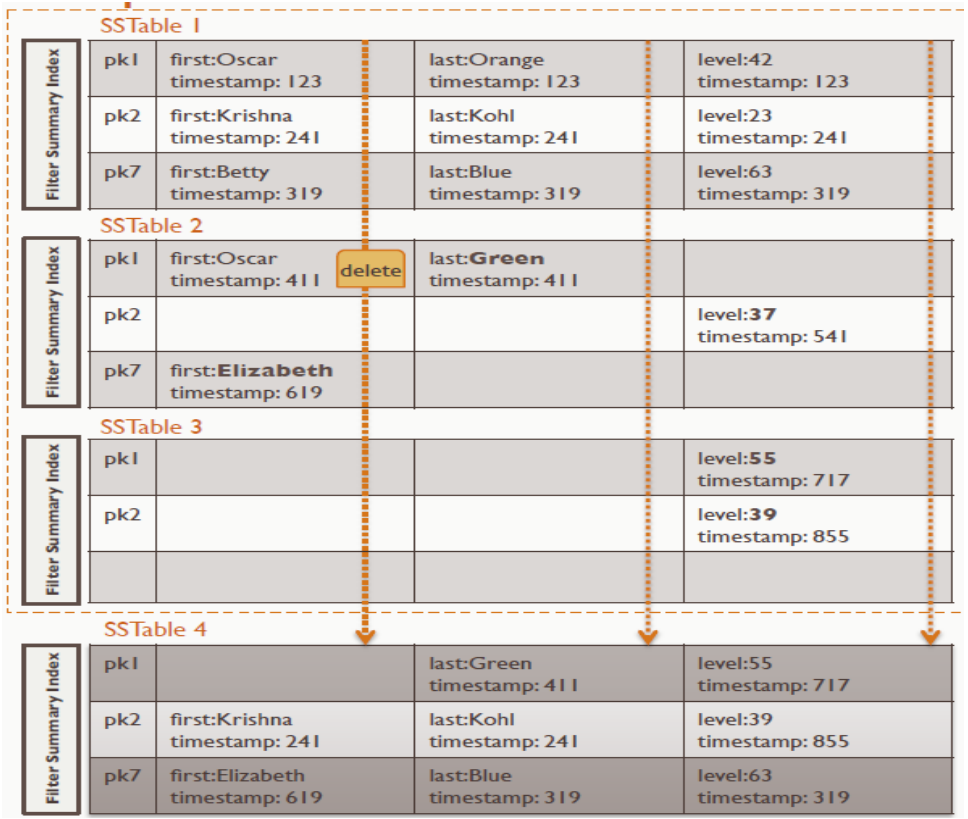
Qu’est ce que la compaction ?

- C’est un processus, critique et périodique de maintenance des SSTables qui:
  - fusionne les colonnes des clés de partitions les plus récentes
  - Évince les colonnes de partition supprimées ou avec un TTL à zéro.
  - Crée une nouvelle SSTable
  - Reconstruit les fichiers de ‘partition index’ et de ‘Partition Summary’
  - Supprime les anciennes SSTables.



Qu’est ce que la compaction ? (2)

- C’est un processus efficace car:
  - Les SSTables sont déjà nativement triées par clé de partition.
  - Pas besoin d’accès disques aléatoires supprimées ou avec un TTL à zéro.
- C’est un processus nécessaire car:
  - Les SSTables sont des fichiers immutables donc les updates ont tendances avec le temps, à fragmenter les données.
  - Les suppressions sont enregistrées et doivent être régulièrement nettoyées.



Comment la compaction impact les lectures et l'espace disque?

- Pendant la compaction:
  - Les I/O disques et l'espace disque consommé augmentent
  - Les performances des lectures 'off-cache' peuvent être impactées.
- Après ma compaction:
  - Les performances en lecture augmentent puisque moins de SSTables sont lues pour les lecture 'off-cache'
  - L'espace disque consommé diminue puisque les anciennes SSTables sont supprimées

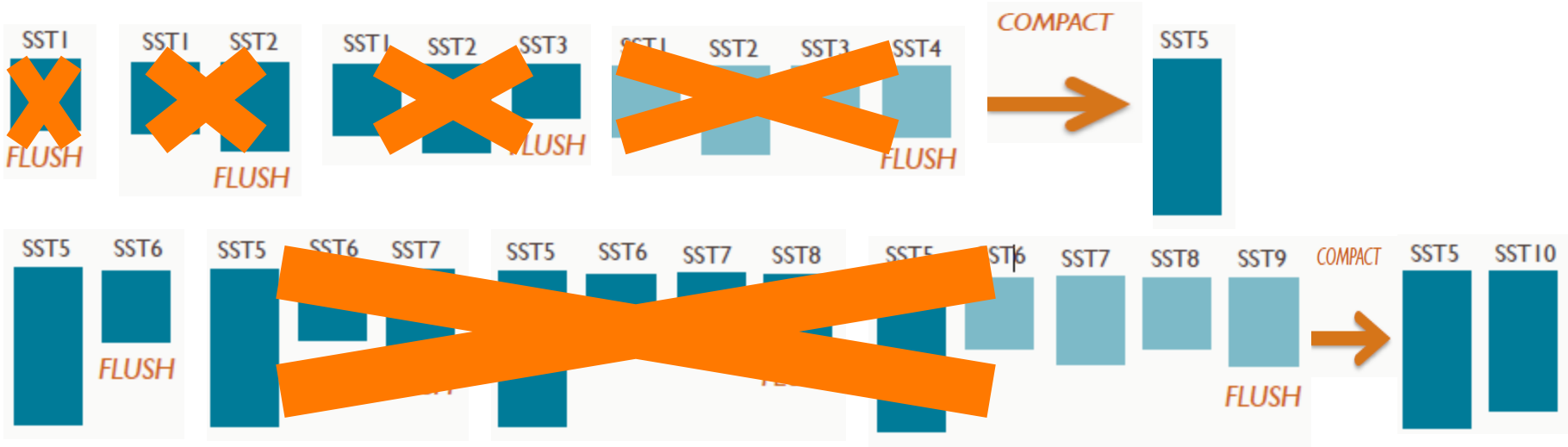
Quelles sont les différentes stratégies de compaction?

- Compaction :
  - C’est une propriété de chaque table qui définie quand et comment déclencher la compaction.
- Les stratégies disponibles sont :
  - Size-Tiered (défaut) : la compaction est déclenchée lorsque le nombre de SSTables atteint un seuil.
  - Leveled: les SSTables sont regroupées par plage de taille similaire et compactées par niveau.
  - Time-Windows : les données des SSTables écrites dans la même fenêtre temporelle sont compactées ensemble.

```
CREATE TABLE performer (
  first text PRIMARY KEY,
  last text,
  level text
)
WITH compaction = {'class' : '<strategy>', <params>};
```

Qu’est ce que la ‘Size-Tiered Compaction’ (STC) ?

- Compacte un ensemble de SSTables de taille similaire en une SSTable plus grande:

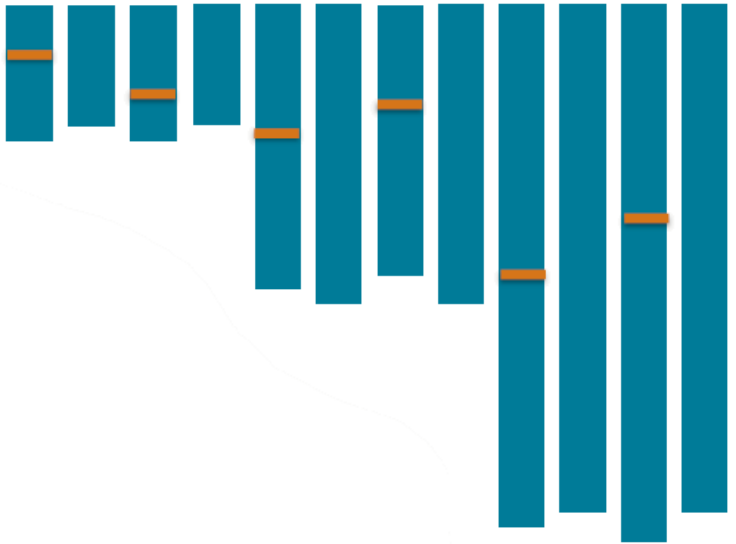



```
ALTER TABLE performer

WITH compaction =
 {'class' : 'SizeTieredCompactionStrategy', <params>};
```

Qu’est ce que la ‘Size-Tiered Compaction’, quand l’utiliser?

- Les implications de cette compaction:
  - Chaque compaction est rapide car assez peu de SSTables sont compactées en une fois
  - Successivement des SSTables plus grandes
  - Temps de réponse en lecture variant pour les partitions fréquemment m.a.j car plus de SSTables à parcourir du fait de la dispersion.
  - Possible perte d’espace si des ‘tombstones’ sont mergées dans d’autres SSTables.
  - Nécessite un espace disque au moins 2 fois égal à l’espace de la plus grosse table.
  - Ce type de compaction est plutôt réservé au profil de charge de type ‘écriture intensive’ et ‘time series’.



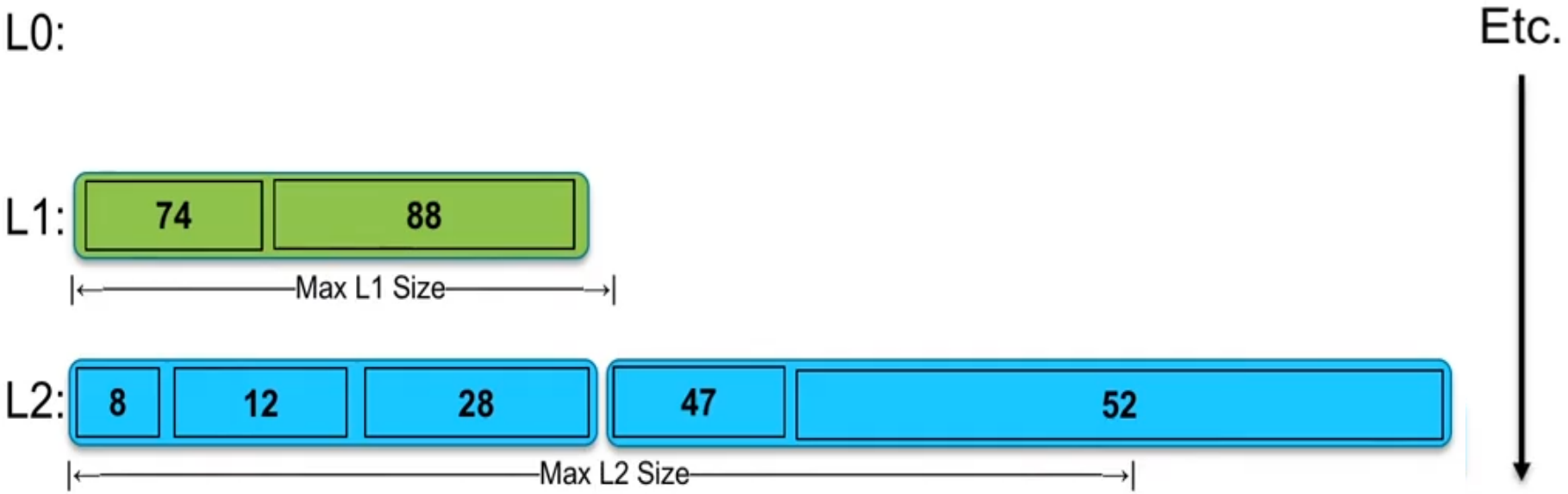
Qu’est qu’une ‘full compaction’ et quand l’utiliser?

- C’est une opération de maintenance, pas une stratégie!
  - Utilisable seulement avec les tables en STC
  - Compacte toutes, ou seulement celles indiquées, des SSTables en Une seule SSTable
  - On parle aussi de ‘compaction majeure’
  - Consomme énormément de ressource disque (espace et I/O)
  - La commande : nodetool compact [keyspace] [table]
  - Si aucune tables n’est spécifiée alors toutes seront compactées.
  - NON recommandée en production.

Qu’est ce que ‘leveled compaction’ et quand l’utiliser?

- Compacte des petites SSTable de taille fixe succéssivement jusqu’à

Qu'est ce que 'leveled compaction' (2)

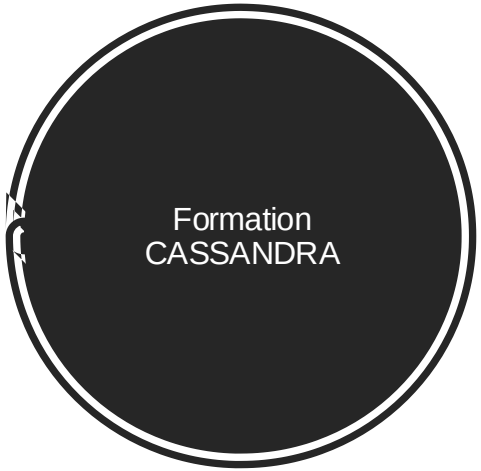


Qu’est ce que ‘leveled compaction’ (3)

- Dans notre exemple le coefficient multiplicateur par niveau était de 2
- En réalité par défaut il est de 10 par niveau.
- Le paramètre donnant la taille maximale d’une sstable est ‘sstable_size_in_mb’, sa valeur par défaut est de 160 Mb.
- Parfois, les sstables peuvent être plus grande pour s’assurer que la dernière partition est écrite complétement dans la même sstables.
- Dans l’exemple, les modèle de données génére de très grande partition, ce qui n’est pas forcément à suivre.
- Plus la granularité de vos partitions sera fine, plus toutes vos sstables seront uniformément de 160 Mb et vous aurez moins de niveau (aussi rapidement..)

Qu’est ce que ‘leveled compaction’ (4)

- La LCS est plus performante pour les profils de charge de type ‘lecture intensives’
  - C.a.d beaucoup plus de lectures que de d’écritures
- Chaque partition n’est présente que dans un fichier sstable d’un niveau donné.
- En général, les lectures porteront sur quelques sstables car:
  - Les partitions se retrouvent regroupées dans une poignée de niveau au fils des compactions
  - 90% des données resident dans le plus bas niveau (du fait de la régle du coefficient multiplicateur de * 10 par niveau descendant)
  - Cette compaction nécessite 11 fois la taille de la plus grande SSTables pour compacter :
    - Une fois pour le niveau le plus haut
    - Et 10 pour les sstables qui se chevauchent dans le niveau suivant.
  - L’espace disque consommé est moins important que pour le STC
  - Mais très consommateur en IO disques, donc attention à la charge globale des nœuds C*
  - Compacte beaucoup plus de sstable en une fois que STC
  - Compacte plus souvent que STC
  - Ne peut pas ingérer les ‘insert’ à haut débit.



- Introduction - Rappel
- Cassandra architecture…
- La réplication des données
- La cohérence des données
- Chemins de lecture & écriture
- Mécanique de maintenace
- **Cassandra & Spark**

Spark

Spark et Cassandra

Les cadres d'usages de Spark avec Cassandra :

Nettoyage, validation, normalisation, transformation de données

Chargement de données
Multi sources

Migration de schéma
Conversion de données

Analytique(jointure, aggrégation, trasformation…)

Qu’est ce que Spark?

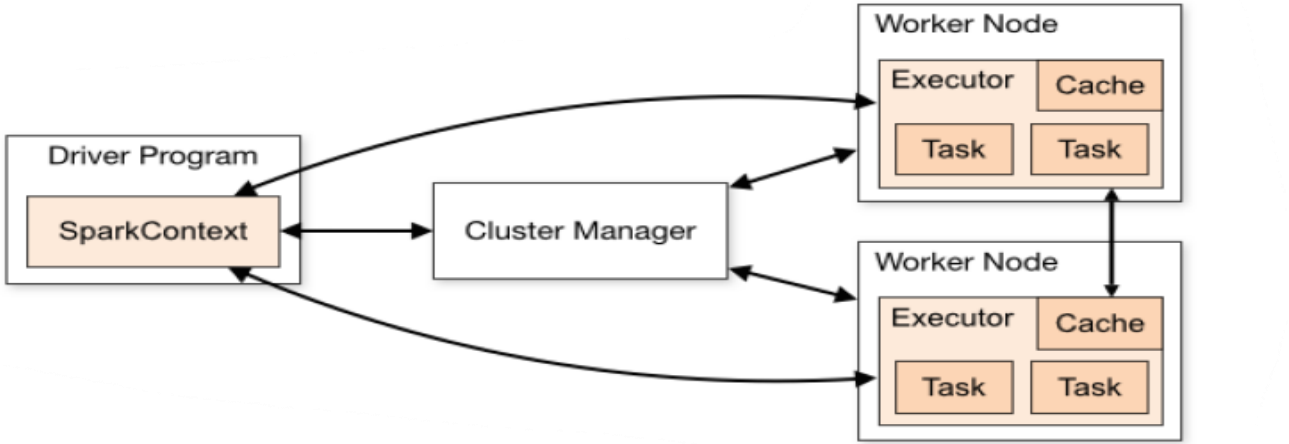
- Un écosystème sous forme de cluster pour processer des calculs qui est :
  - Résistant au pannes, car repose sur un cluster
  - Rapide car en mémoire quand bien dimmensionné
  - Convient aussi bien au temps réel qu’au batch
  - Scalable, de part sa distribution en parallèle
- Spark fonctionne au dessus d’un gestionnaire de ressource:
  - Local (pour faire du dev)
  - Standalone (sur server dédié de préférence)
  - Avec YARN (comme gestionnaire de ressource)
  - MESOS, etc…
- Fourni une API de programmation fonctionnelle:
  - Scala : langage natif
  - Java
  - Python

Spark et Cassandra:

- Spark peut être relié à différentes base de données dont :
  - Cassandra
  - ElasticSearch
  - MongoDB
  - Couchbase
  - Hbase
- Par des connecteurs dédiés.

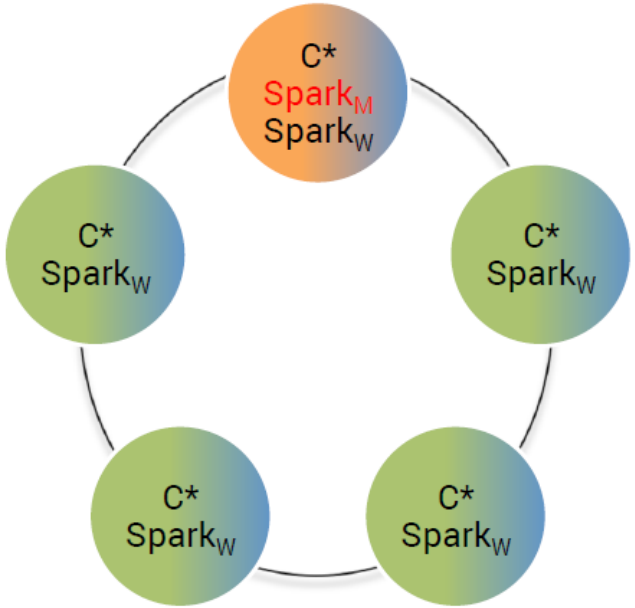
Spark architecture interne simplifiée

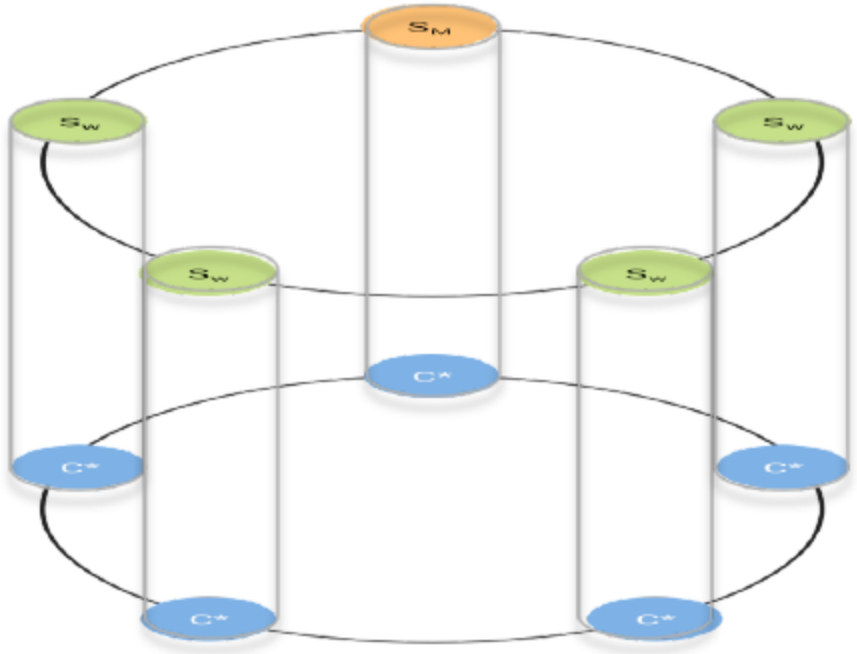
- Un programme Spark est:
  - Une ‘driver program’ appelé aussi ‘master’ qui contient un ‘spark context’ (en quelques sorte les métadata nécessaire à la connexion avec la source de données par exemple)
  - Execute en parallèle des opérations sur des ‘datasets’ sur des nœuds dits ‘worker’
  - Le ‘cluster manager’ décide de l’allocation des ressources aux ‘workers’
  - Les applications lancées ne sont pas distribuées mais chaque opérations le sont!!



Déploiement de cluster Spark

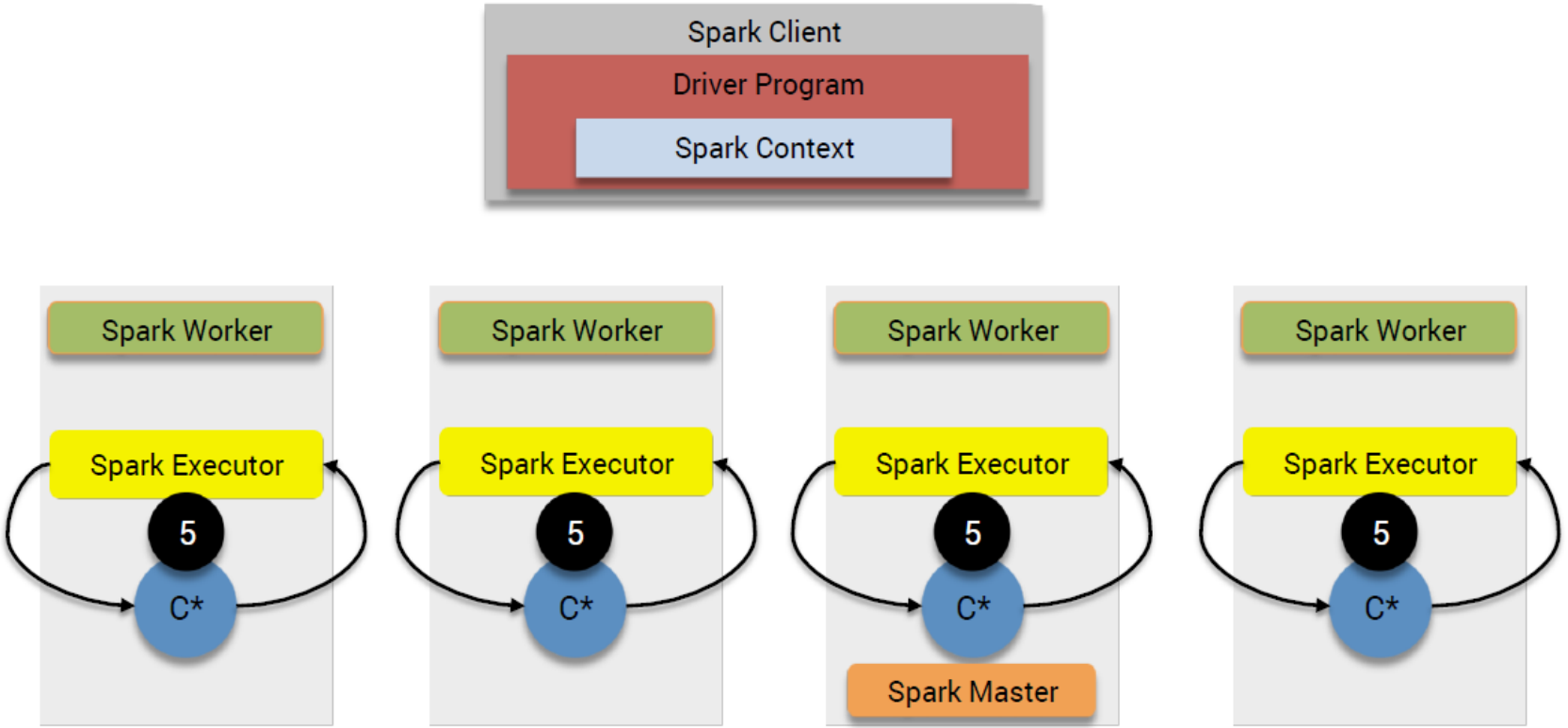
- Cluster de type ‘standalone’ :





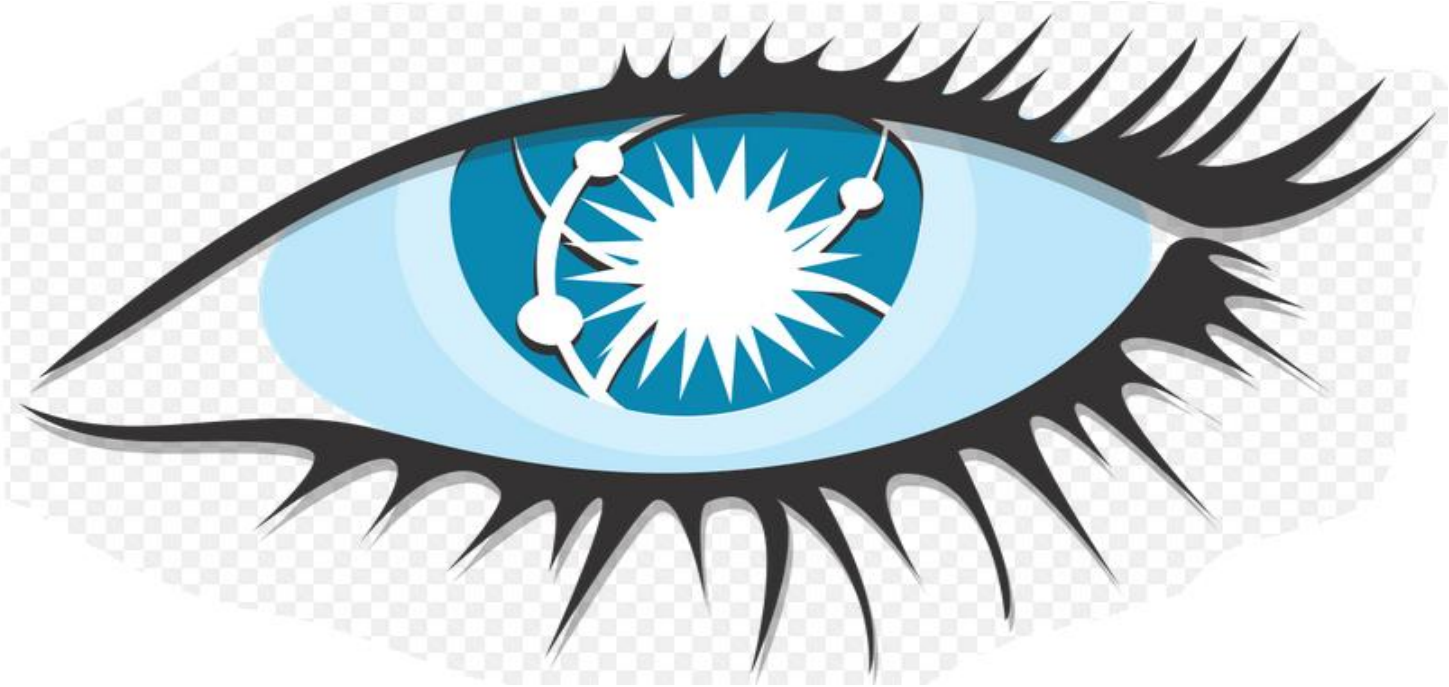
Les échanges entre Cassandra et Spark (les jobs)

1 processe C* ⇔ 1 worker Spark



DES QUESTIONS?

Merci



ANNEXE