esprit
Ecole Supérieure Privée
d'Ingénierie et de Technologies

Chp7 – Hbase

2016-2017

Mme Ines Slimene
ines.slimen@esprit.tn

Plan module

- Introduction
- Écosystème Hadoop
- HDFS
- MapReduce
- Langages de requête Hadoop :Pig, Hive
- SGBDNR
  - Différences entre une BDNR et une BD relationnelle
  - Typologies des BD non relationnelles
- **Etude d'un SGBDNR : HBase**

Plan

- Introduction
- Quand utiliser Hbase ?
- Quand ne pas utiliser Hbase ?
- Modèle de données
- Déploiement
- Création des tables
- Gestion des données

Introduction

- Base de données orientée colonne, connue par “Hadoop Database”
  - Supporte des opérations CRUD en temps réel.
- Distribuée, conçue pour stocker des tables de grand volume.
  - Billions de lignes et millions de colonnes
- Open-source, écrit en Java
- Un type de base de données “NoSQL”

Introduction

- Scalabilité horizontale
  - Shard automatique
- Forte cohérence de données
- Intégration avec le framework MapReduce
- Basé sur le concept Bigtable de google

Quand utiliser HBase

- Avoir suffisamment de hardware
  - Minimum 5 nœuds
- Avoir, pour chaque bloc, 3 réplications
- HBase est consommateur d'espace mémoire et de CPU.

Quand ne pas utiliser Hbase

- Utilisation similaire a SGBDR
  - Analyse relationnelle
  - 'group by', 'join', et 'where column like', etc....
- Accès par Recherche textuelle

Modèle de données HBase

- Les données sont stockées dans des tables
- Les tables contiennent des lignes
  - Les lignes sont référencées par une clé
- Les lignes sont composées de colonnes groupées par famille
- Les données sont stockées dans des cellules
  - Identifiées par ligne*colonne-famille*colonne

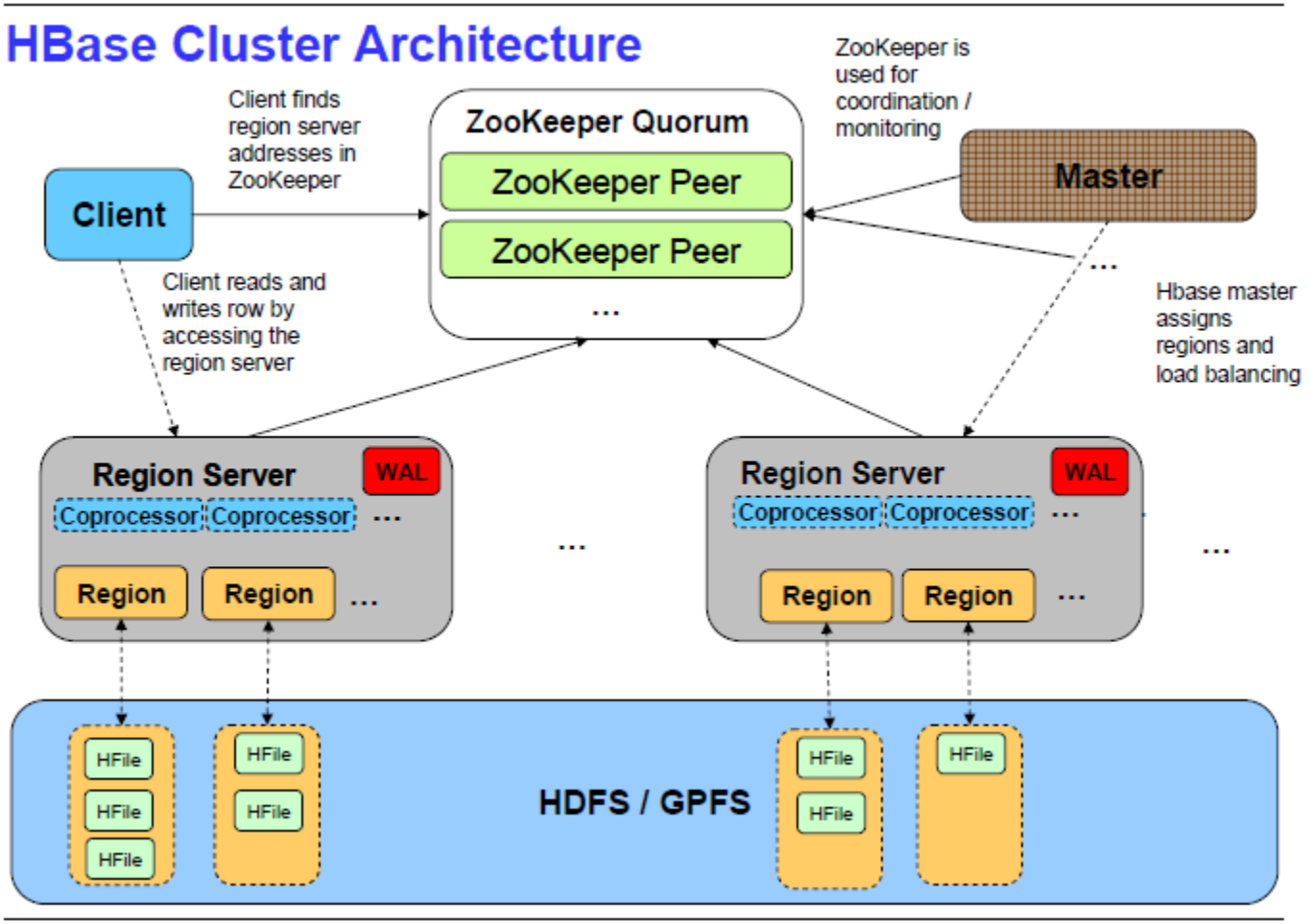
Familles HBase

- **Les lignes sont groupées par famille**
  - Définies par “family:column”
    - Exemple “user:first_name”
- **Définition d’une famille**
  - Stockées ensemble dans un fichier HFile/StoreFile
  - Créée avec les tables, peut être rarement ajoutée ou modifiée
  - nombre limité de famille par table
    - Contrairement au colonnes dont le nombre peut atteindre l’ordre des millions

Hbase : versions

- Pour chaque cellule plusieurs versions sont sauvegardées
  - 3 par défaut
- Les versions sont stockées dans un ordre décroissant du timestamp
- On peut spécifier le nombre de versions qui doivent être gardées

Hbase : Déploiement



Régions

- **Région**
  - Ensemble de lignes de table
  - Les données d'une région sont automatiquement réparties par le serveur région quand elles atteignent une taille bien déterminée.
  - Contient les tables, exécute les lectures, enregistre les écritures
  - Périodiquement, un load balancer déplacera les régions dans les clusters pour équilibrer les charges
  - Les Clients interagissent directement avec eux pour les opérations de lecture/écriture

Master

- **Master**
  - Ne stocke ni lit les données.
  - Responsable de la coordination entre les Régions Serveurs
  - Détecte l'état des Régions Serveurs,
  - Assigne les Régions aux Régions Serveurs
  - Utilise Zookeeper pour distribuer et coordonner les services
  - Possibilité de plusieurs serveurs Master :
    - 1 primaire et plusieurs serveurs backup

Zookeeper

- **Zookeeper**
- Produit Apache open source et une partie de l'ecosystème Hadoop
- Composant primordial pour HBase
- S'assure qu'il n'y a qu'un seul Master en marche
- Enregistre les Régions Serveurs
- Gère les erreurs des Régions et du serveur Master
- allocation des régions

HBase : Shell

- Démarrage Hbase
  - hbase shell
- Langage de définition de données (LDD)
  - alter, create, describe, disable, drop, enable, exists, is_disabled, is_enabled, list
- Langage de manipulation de donnée (LMD)
  - count, delete, get, get_counter, put, scan, truncate
- Administration de cluster
  - balancer, close_region, move, split, unassign, start_replication, stop_replication….

1: Creation de Table

- **Créer une table 'Blog' avec le schéma suivant :**
  - 2 familles
    - 'info' avec 3 colonnes : 'title', 'author', et 'date'
    - 'content' avec une colonne : 'post'

Blog		
Family:	info:	Columns: title, author, date
	content:	Columns: post

1: Creation de Table

- Plusieurs options de création de table.
  - hbase> create 't1', {NAME => 'f1', VERSIONS => 5}
  - hbase> create 't1', {NAME => 'f1'}, {NAME => 'f2'},{NAME => 'f3'}
- Exemple :

hbase> create 'Blog', {NAME=>'info'}, {NAME=>'content'}

2: insertion de données

- hbase> put 'table', 'row_id', 'family:column', 'value‘
- Exemple :
  - put 'Blog', 'Matt-001', 'info:title', 'Elephant'
  - put 'Blog', 'Matt-001', 'info:author', 'Matt'
  - put 'Blog', 'Matt-001', 'info:date', '2009.05.06'
  - put 'Blog', 'Matt-001', 'content:post', 'Do elephants like monkeys?'

3. Accès aux données - count

- **Accès aux données**
  - count: nombre total d'enregistrement
  - get: afficher une seule ligne
  - scan: afficher une liste de ligne
- **Count**
  - hbase> count 'table_name'
  - Parcours toute la table.
  - Peut être lent pour une table volumineuse

3. Accès aux données - count

- hbase> **count 'Blog', {INTERVAL=>2}**

Current count: 2, row: John-005

Current count: 4, row: Matt-002

5 row(s) in 0.0220 seconds

- hbase> **count 'Blog', {INTERVAL=>1}**

Current count: 1, row: Bob-003

Current count: 2, row: John-005

Current count: 3, row: Matt-001

Current count: 4, row: Matt-002

Current count: 5, row: Michelle-004

3. Accès aux données - get

- **Get : affiche une seule ligne de la table**
  - hbase> get 'table', 'row_id'
- **Afficher des colonnes bien déterminées**
  - hbase> get 't1', 'r1', {COLUMN => 'c1'}
  - hbase> get 't1', 'r1', {COLUMN => ['c1', 'c2', 'c3']}
- **Afficher pour un timestamp bien déterminé**
  - hbase> get 't1', 'r1', {COLUMN => 'c1', TIMESTAMP => ts1}
- **Afficher plus qu'une version**
  - hbase> get 't1', 'r1', {VERSIONS => 4}

3. Accès aux données - get

- hbase> **get 'Blog', 'Matt-001',{COLUMN=>'info:date', VERSIONS=>2}**

info:date timestamp=1326071670471, value=1990.07.08
info:date timestamp=1326071670442, value=1990.07.07
2 row(s) in 0.0300 seconds

- hbase> **get 'Blog', 'Michelle-004',{COLUMN=>'info:date'}**

info:date timestamp=1326071670471, value=1990.07.08
1 row(s) in 0.0190 seconds

3. Accès aux données - Scan

- **Afficher toutes les lignes de la table**
  - hbase> scan 'table_name'
- **Limiter le nombre de données affichées**
  - hbase> scan 'table_name', {LIMIT=>1}
- **Afficher un ensemble de données**
  - hbase> scan 'Blog', {STARTROW=>'startRow', STOPROW=>'stopRow'}
  - Start row est inclus, stop row est exclut
  - Exemple :

```
hbase> scan 'Blog', {COLUMNS=>'info:title',
                     STARTROW=>'John',
                     STOPROW=>'Michelle'}
```

4: Gestion des données

- La commande Put ajoute la ligne si elle n'existe pas.
- Put met à jour si la ligne existe.
- Pas une vrai mise à jour
  - Ajouter une nouvelle version pour la cellule
  - N versions conservées par cellule
  - Le nombre de version est configuré par famille lors de la création :

```
hbase> create 'table', {NAME => 'family', VERSIONS => 7}
```

4: Gestion des données

- hbase> **put 'Blog', 'Michelle-004', 'info:date', '1990.07.06'**
- hbase> **put 'Blog', 'Michelle-004', 'info:date', '1990.07.07'**
- hbase> **put 'Blog', 'Michelle-004', 'info:date', '1990.07.08'**
- hbase> **get 'Blog', 'Michelle-004',{COLUMN=>'info:date', VERSIONS=>3}**
- hbase> **get 'Blog', 'Michelle-004',{COLUMN=>'info:date', VERSIONS=>2}**

5: Suppression des enregistrements

- **Suppression des cellules en définissant la table l'identifiant et la colonne**
  - delete 'table', 'rowId', 'column'
  - Supprimer toutes les versions d'une ligne
- **On peut ajouter le timestamp pour supprimer les versions qui lui sont antérieur**
  - delete 'table', 'rowId', 'column', timestamp

5: Suppression des enregistrements

- hbase> **get 'Blog', 'Matt-001', 'info:date'**

1 row(s) in 0.0200 seconds

- hbase> **delete 'Blog', ' Matt-001', 'info:date'**

0 row(s) in 0.0180 seconds

- hbase> **get 'Blog', ' Matt-001', 'info:date'**

COLUMN CELL

0 row(s) in 0.0170 seconds

5: Suppression des enregistrements

- hbase> **get 'Blog', 'Michelle-004',{COLUMN=>'info:date', VERSIONS=>3}**

COLUMN CELL

info:date timestamp=1326254742846, value=1990.07.08

info:date timestamp=**1326254739790**, value=1990.07.07

info:date timestamp=1326254736564, value=1990.07.06

3 row(s) in 0.0120 seconds

- hbase> **delete 'Blog', 'Michelle-004', 'info:date', 1326254739791**

0 row(s) in 0.0150 seconds

- hbase> **get 'Blog', 'Michelle-004',{COLUMN=>'info:date', VERSIONS=>3}**

COLUMN CELL

info:date timestamp=1326254742846, value=1990.07.08

1 row(s) in 0.0090 seconds

6: Drop table

- On doit désactiver la table avant la suppression
- Exemple :
  - hbase> **list**

  Blog

  1 row(s) in 0.0120 seconds
  - hbase> **disable 'Blog'**

  0 row(s) in 2.0510 seconds
  - hbase> **drop 'Blog'**

  0 row(s) in 0.0940 seconds
  - hbase> **list**

  0 row(s) in 0.0200 seconds