esprit
Ecole Supérieure Privée
d'Ingénierie et de Technologies

Chp5 – Langages de requête Hadoop-Pig

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
- Pig, Hive –Similarités
- Pig, Hive –Différences
- Pig
  - Cas d'utilisation
  - Concepts
  - Commandes
  - Exemples

Pig vs MapReduce

- Pig fournit un langage de haut niveau conçu pour formuler des programmes d'analyse de données.
- converti en MapReduce et exécuté dans un cluster Hadoop.
- Il peut être utilisé par :
  - Analystes
  - Scientifiques de données
  - Statisticiens
  - Etc...
- Initialement mis en œuvre par Yahoo! pour permettre aux analystes d'accéder aux données

Pig : cas d'utilisation

- **Extract Transform Load (ETL)**
  - Ex: traitement d'un grand volume de fichiers log
- **Recherche d'information**
  - Exemple : fichier d'audit
  - Le schéma peut-être inconnu ou incohérent

Pig : composants

- **Pig Latin**
  - Langage basé sur les commandes
  - Conçu spécifiquement pour la transformation de données et l'expression des flux
- **Environnement d'exécution JVM**

Pig : Modes d'exécution

- **Mode local**
  - S'exécute sur une JVM
  - Fonctionne exclusivement avec le système de fichiers local

    pig –x local
- **Mode Hadoop**
  - Aussi connu par mode MapReduce
  - Pig transforme Pig Latin en des jobs MapReduce et les exécute dans un cluster

    pig –x mapreduce

Pig Latin : Concept

- **Concept**
  - Champ – partie des données
  - Tuple – ensemble de champs. Représenté par “(“ et “)”
    - (10.4, 5, word, 4, field1)
  - Bag – collection de tuples. Représenté par “{“ et “}”
    - { (10.4, 5, word, 4, field1), (this, 1, blah) }
- **Similaire aux bases de données relationnelles**
  - Bag ⇔ table
  - Tuple ⇔ ligne d’une table
  - Les tuples d’un même bag peuvent ne pas avoir le même nombre de colonnes.

Pig Latin : Commandes

- Trois étapes pour un programme Pig typique :
  - LOAD :
    - Charger les données à partir de HDFS.
  - TRANSFORM
    - Traduire en un ensemble de taches map et reduce
    - Operateurs relationnels : FILTER, FOREACH, GROUP, UNION, DISTINCT, ORDER …etc.
  - DUMP or STORE
    - Afficher les résultats sur l'écran ou les stocker dans un fichier

Commande LOAD

LOAD 'data' [USING function] [AS schema];

- **data – le nom du fichier ou du chemin**
- **USING – définir la fonction load à utiliser**
  - Par défaut utilise la fonction PigStorage qui divise chaque ligne en champs en utilisant un délimiteur.
    - Le délimiteur par défaut est tab (‘\t’)
- **AS – donne un schéma aux données en entrée**
  - Donne des noms et des types aux champs crées

Commandes DUMP et STORE

- Les requêtes ne sont exécutées qu'à la suite de l'exécution des commandes DUMP ou STORE
- Pig analyse et valide les requêtes mais ne les exécute pas.
- DUMP – affiche les résultats sur l'écran
- STORE – enregistre le résultat (typiquement dans un fichier)

Exemple 1:

- grunt> student = LOAD 'ch5pig/student_details.txt' USING PigStorage(',') as (id:int, firstname:chararray, lastname:chararray,age:int, phone:chararray, city:chararray);
- grunt> order_by_data = ORDER student BY age DESC;
- grunt> limit_data = LIMIT student 4;
- grunt> distinct_data = DISTINCT student;
- grunt> STORE student INTO 'pig_Output';

Exemple 1– outils de diagnostique

- **Afficher le schéma (les champs, noms et types, de la relation).**
  - grunt> DESCRIBE student;
- **Afficher le plan d'exécution**
  - grunt> EXPLAIN student;
- **Illustrer comment pig transforme les données.**
  - grunt> ILLUSTRATE student;

Exemple 2: Exécution de script

```
/* drivers */
--chargement des données
drivers = LOAD 'ch5pig/drivers.csv' USING PigStorage(',');
--selection des drivers dont l'id > 20
raw_drivers = FILTER drivers BY $0>20;
-- affichage des colonnes driverId et name
drivers_details = FOREACH raw_drivers GENERATE $0 AS driverId, $1 AS name;
--calculer total des nombre d'heure de travail par driverId
grp_logged = GROUP timesheet_logged by driverId;
sum_logged = FOREACH grp_logged GENERATE group as driverId,
SUM(timesheet_logged.hours_logged) as sum_hourslogged;
--Affichage du nombre d'heure de travail par nom de chauffeur
join_sum_logged = JOIN sum_logged by driverId, drivers_details by driverId;
join_data = FOREACH join_sum_logged GENERATE $3 as name, $1 as hours_logged;
Store join_data INTO ‘output’;
```

Exemple 2: Exécution de script

- Enregistrer le script sous le nom drivers.pig
- Exécuter le script avec la commande : pig –x mapreduce drivers.pig