DI PLACIDO Anna
M2 Informatique

LSDS – RAPPORT DE PROJET

GITHUB : Annadip5/LSDS2

Préparation

Les commande et les détails de l'utilisation sont dans le redme.md

Les images d'exécutions et de résultats sont dans le dossier /images et le readme.md

Projet

Le projet consiste à effectuer diverses opérations sur les données de films et d'évaluations, en utilisant des jobs Hadoop pour analyser et extraire des informations pertinentes.

HighestRatedMoviePerUser

Le programme `HighestRatedMoviePerUser` est conçu pour déterminer quel est le film le mieux noté par chaque utilisateur à partir des données d'évaluation (ratings). Il utilise Hadoop MapReduce pour traiter les données et obtenir un résultat sous forme de liste des films ayant les meilleures évaluations par utilisateur.

Fonctionnalité principale :

- Le programme prend deux arguments en entrée :
- Le fichier d'entrée `ratings.csv` contenant les évaluations des utilisateurs.
- Le répertoire de sortie `highestMovieRatedPerUser` où les résultats seront stockés.

Fonctionnalité du Mapper :

- Il parcourt chaque ligne du fichier et extrait le `userId` (l'identifiant de l'utilisateur) et les informations de film (`movieId`, `rating`).
- Si la ligne est incorrecte ou corrompue (par exemple, un format invalide), elle est ignorée et un avertissement est loggé.
- Les paires clé-valeur sont envoyées au Reducer :
- **Clé** : `userId` (identifiant de l'utilisateur).
- **Valeur** : Une chaîne contenant `movieId` et `rating` séparés par une virgule.

Fonctionnalité du Reducer :

- Le Reducer reçoit toutes les évaluations des films pour un utilisateur sous forme d'une liste (chaîne de caractères contenant `movieId,rating`).
- Il compare les évaluations et garde celle avec la meilleure note.
- Le résultat final est une paire clé-valeur avec :
- **Clé :** `userId` (identifiant de l'utilisateur).

- **Valeur :** `movieId` du film avec la meilleure note.

JoinHighestRatedMovie

Le programme `JoinHighestRatedMovie` utilise Hadoop MapReduce pour effectuer une jointure entre les résultats de la question 1, qui contiennent les films les mieux notés par chaque utilisateur, et les informations détaillées sur les films (comme le nom du film). L'objectif est de lier chaque `userId` aux films qu'il a évalués, en affichant le nom de ces films.

Fonctionnalité principale :

- Le programme prend trois arguments en entrée :
- Le fichier contenant les résultats de la question précédante
- Le fichier contenant les informations des films (`movies.csv`).
- Le répertoire de sortie pour stocker les résultats.
- Le job Hadoop est créé avec deux Mappers (`MoviesMapper` et `RatingsMapper`) et un Reducer (`JoinReducer`).
- L'opération de jointure est effectuée par le `Reducer`, qui associe le nom du film avec les id des films.

Fonctionnalité du mapper:

- Le Mapper lit le fichier des films et sépare les champs avec une virgule.
- Si l'entrée est valide (non-entête), il extrait `movieId` et le nom du film.
- La sortie est une paire clé-valeur où la **clé** est `movieId` et la **valeur** est le nom du film, préfixée par `"M:"`.

Fonctionnalité du RatingsMapper:

- Le Mapper lit les évaluations des utilisateurs et les sépare en fonction des tab.
- Il extrait `userId` et `movieId`.
- La sortie est une paire clé-valeur où la **clé** est `movieId` et la **valeur** est l'ID de l'utilisateur, préfixée par `"U:"`.

Fonctionnalité du Reducer (réalise la jointure) :

- Le Reducer reçoit toutes les paires avec la même clé `movieId` et parcourt les valeurs associées à cette clé.
- Pour chaque valeur venant de `MoviesMapper` (préfixée par `"M:"`), le nom du film est extrait.
- Pour chaque valeur venant de `RatingsMapper` (préfixée par `"U:"`), l'ID de l'utilisateur est extrait.
- Enfin, une sortie est générée sous la forme de :
- **Clé** : `movieId`.

- **Valeur** : Liste des noms de films, séparés par des espaces, pour chaque utilisateur ayant évalué ce film.

Count likes per film + group by like count

Le programme `GroupMoviesByLikeCount` est un job Hadoop qui s'exécute en deux étapes pour analyser et grouper les films par nombre de "likes". Voici les différentes étapes :

Job 1 : Compter les likes par film

- Mapper (`UserMovieMapper`) : Il lit les données d'entrée (format `userId\tmovieName`) et associe un film (`movieName`) à chaque occurrence.
- Reducer (`MovieLikeCounterReducer`) : Il agrège les valeurs (nombre de fois qu'un film a été aimé) pour chaque film et génère un total de "likes" pour chaque film.

Job 2 : Grouper les films par nombre de likes

- Mapper (`GroupByLikeCountMapper`): Il lit les sorties du job 1 et associe chaque film à son nombre de "likes" pour les regrouper.
- Reducer (`GroupByLikeCountReducer`) : Il regroupe les films en fonction du nombre de "likes" et génère une liste de films pour chaque nombre de "likes".

Résultat

Le programme génère un fichier de sortie qui contient, pour chaque nombre de "likes", les films correspondants.