L'Optimisation Chez Apache Spark™

APACHE SPARK

DONNEES DISTRIBUEES

Spark est un framework open source de **traitement de données distribué**. Il s'agit d'un ensemble d'outils et de composants logiciels développés pour fournir une alternative rapide et généralisée à MapReduce, le modèle de traitement de données associé à Apache Hadoop.

GRANDE ECHELLE

Spark est conçu pour le traitement de données à grande échelle, permettant de traiter des **volumes massifs** de données en les distribuant sur **un cluster de machines**.

NOMBREUX SECTEURS

L'utilisation de Spark est répandue dans de nombreux secteurs pour des applications telles que l'analyse de données, le traitement de flux en temps réel, le machine learning et plus encore. Sa polyvalence et sa rapidité en font un choix très populaire chez les développeurs.

EXEMPLES

OPTMISATION DES REQUÊTES : Imaginons que l'on dispose d'une base donnée contenant les toutes les informations associées à des utilisateurs et que l'on souhaite récupérer le nom de tous ceux âgés de plus de 21 ans, nous pourrions le faire de deux manières différentes.
Dans cet exemple, la version optimisée sélectionne uniquement les **colonnes nécessaires** avant d'appliquer le filtre. Cela **réduit la quantité de données traitées**, améliorant ainsi les performances.

catalystOptimizer.py

```sql
# Non optimisé
SELECT city, AVG(age) AS average_age
FROM users
WHERE city = 'Paris'
GROUP BY city;

# Optimisé
SELECT city, AVG(age) AS average_age
FROM users
GROUP BY city
HAVING city = 'Paris';
```

optimisationRequetes.py

```python
# Non optimisé
df_non_optimized = df.select("*").filter(df["age"] > 21)

# Optimisé
df_optimized = df.select("name").filter(df["age"] > 21)
```

CATALYST OPTMIZER : dans un premier temps nous filtrons les utilisateurs de la ville de Paris avant de calculer la moyenne d'âge. Il est possible de réécrire cette requête de manière plus efficace en effectuant la filtration après le calcul de la moyenne. Cela peut réduire le nombre d'enregistrements à considérer lors du calcul de la moyenne et donc être plus performant.
Ce genre d'optimisation peut facilement être repéré par le Catalyst Optimizer.

METHODES D'OPTMISATION

01 — PLACEMENT MEMOIRE
optimisation des ressources, placement statique, mémoire cache et memory spilling

02 — PARALLELISME
efficacité grâce au traitement en parallèle, division des données en partitions

03 — LAZY EVALUATION
Retarder l'évaluation des transformations sur les données, plan d'exécution logique (DAG)

04 — OPTIMISATION
Sélection des éléments nécessaires, partitionnement des données, stockage des résultats

05 — CATALYST OPTIMIZER
optimiseur de requêtes intégré à Apache Spark, analyse et transforme le plan de requête logique

istic Informatique Électronique

Master II – Ingénierie Logiciel
Année 2023 – 2024

FOUILLEN Mathis, GOMBERT Gwenaël
RASEMONT Florian, RIBEIRO GOMES Lise

Université de Rennes