Base de données NOSQL et Big Data

Projet semestrielle: MapReduce

Introduction :

L'objectif de cette analyse est d'explorer les données historiques sur les prix des actions de 500 grandes entreprises au cours des cinq dernières années. Les données sont extraites de Yahoo Finance et incluent des métriques quotidiennes telles que l'ouverture, la fermeture, le volume, etc. L'analyse vise à fournir des insights pour les investisseurs, les chercheurs et les passionnés d'apprentissage automatique.

Étape 1: Exploration des Données

Dans cette étape, nous avons chargé le jeu de données à l'aide de la bibliothèque Pandas en Python. Nous avons affiché les premières lignes du jeu de données pour comprendre sa structure et vérifié la présence de valeurs manquantes et effectué des statistiques de base.

```python
import pandas as pd
import numpy as np

# Charger le jeu de données
df = pd.read_csv('stock_details_5_years.csv')

# Afficher les premières lignes du jeu de données
print(df.head())

# Vérifier les valeurs manquantes
print(df.isnull().sum())

# Statistiques de base
print(df.describe())
```

Étape 2: Nettoyage des Données

Dans cette étape, nous avons supprimé les lignes avec des valeurs manquantes, éliminé les doublons et converti la colonne de date au format datetime. Nous avons également assuré que les types de données étaient appropriés.

```python
# Supprimer les lignes avec des valeurs manquantes
df = df.dropna()

# Vérifier et supprimer les doublons
df = df.drop_duplicates()

# Convertir la colonne de date au format datetime
df['Date'] = pd.to_datetime(df['Date'])

# Assurer que les types de données sont appropriés
df['Volume'] = df['Volume'].astype(int)

# Sauvegarder le jeu de données nettoyé
df.to_csv('cleaned_dataset.csv', index=False)
```

Étape 3: Visualisation des Données

Nous avons utilisé la bibliothèque Matplotlib pour visualiser les prix des actions d'une entreprise spécifique au fil du temps.

```python
import matplotlib.pyplot as plt

# Visualiser les prix des actions pour une entreprise spécifique
company_data = df[df['Company'] == 'AAPL']  # Remplacer 'AAPL' par le symbole de l'entreprise souhaitée
plt.plot(company_data['Date'], company_data['Close'])
plt.title('Prix des actions au fil du temps')
plt.xlabel('Date')
plt.ylabel('Prix de clôture')
plt.show()
```

▪ MapReduce avec Hadoop

Nous avons conçu une tâche MapReduce pour calculer le prix de clôture moyen par entreprise sur l'ensemble du jeu de données. Le script MapReduce est écrit en Java et comprend deux parties principales : le mappage et la réduction. Le mappage extrait les paires clé-valeur à partir des données, tandis que la réduction agrège ces paires pour calculer la moyenne.

- **Mapper.py:**

```python
#!/usr/bin/env python

import sys

# Entrée : Date,Open,High,Low,Close,Volume,Dividends,Stock Splits,Company
for line in sys.stdin:
    # Divise la ligne d'entrée en jetons
    tokens = line.strip().split(",")

    # Vérifie si la ligne a suffisamment de jetons
    if len(tokens) >= 9:
        company = tokens[8]
        closing_price = float(tokens[4])

        # Sortie paire clé-valeur : (company, closing_price)
        print("{}\t{}".format(company, closing_price))

```

- **Reducer.py:**

```python
#!/usr/bin/env python

import sys

current_company = None
total_closing_price = 0.0
count = 0

# Lit l'entrée à partir de la saisie standard
for line in sys.stdin:
    # Divise la ligne d'entrée en clé et valeur
    company, closing_price = line.strip().split("\t")
    closing_price = float(closing_price)

    # Vérifie si l'entreprise a changé
    if current_company != company:
        # Résultat de sortie pour l'entreprise précédente
        if current_company:
            average_closing_price = total_closing_price / count
            print("{}\t{}".format(current_company, average_closing_price))

        # Réinitialise les variables pour la nouvelle entreprise
        current_company = company
        total_closing_price = closing_price
        count = 1
    else:
        # Met à jour les variables pour l'entreprise en cours
        total_closing_price += closing_price
        count += 1

# Résultat de sortie pour la dernière entreprise
if current_company:
    average_closing_price = total_closing_price / count
    print("{}\t{}".format(current_company, average_closing_price))

```

▪ MapReduce avec multiprocessing (Python)

Nous avons également implémenté une version MapReduce en utilisant le module multiprocessing en Python. Cela permet de tirer parti du traitement parallèle sur une machine locale.

Étape 4: MapReduce Design

Dans cette étape, nous concevons le processus MapReduce en définissant les fonctions de mappage et de réduction ainsi que la stratégie de division des données en morceaux pour le traitement parallèle.

```python
import matplotlib.pyplot as plt

# Visualiser les prix des actions pour une entreprise spécifique
company_data = df[df['Company'] == 'AAPL']  # Remplacer 'AAPL' par le symbole de l'entreprise souhaitée
plt.plot(company_data['Date'], company_data['Close'])
plt.title('Prix des actions au fil du temps')
plt.xlabel('Date')
plt.ylabel('Prix de clôture')
plt.show()
```

Étape 5: MapReduce Implementation

Dans cette étape, nous mettons en œuvre le processus MapReduce en utilisant le module multiprocessing en Python. Cela permet de tirer parti du traitement parallèle sur une machine locale.

```
# Créer un pool de travailleurs
with Pool(num_chunks) as pool:
    # Phase de map :
    #
 Appliquer la fonction de map à chaque morceau en parallèle
    mapped_results = pool.map(map_function, data_chunks)

    # Phase de réduction :
    # Combinaison des résultats de la phase de map
    final_result = reduce_function(mapped_results)
```

Étape 6: MapReduce Execution

Dans cette étape, nous exécutons le processus MapReduce en affichant le résultat final, qui peut être utilisé pour des analyses plus approfondies.

```
# Display the final result
print(final_result)
```

Conclusion

Ce rapport a couvert toutes les étapes du processus d'analyse des données sur les prix des actions, de l'exploration initiale à l'application de MapReduce avec la version parallèle en utilisant le module multiprocessing en Python. Ces méthodes offrent une approche scalable pour traiter des volumes importants de données. Les résultats peuvent être utilisés pour prendre des décisions éclairées en matière d'investissement, d'analyse de marché et de modélisation prédictive.