Membre du groupe :

Mohamed Attoisse et Diallo Saikou oumar

Comparaison entre l'utilisation de PySpark et Polars pour le traitement des données

1. Introduction

Dans le cadre de l'analyse des données relatives aux accidents de la circulation, deux approches distinctes ont été utilisées :

- **PySpark sur Databricks** (fichier `examen_big_data_spark.py`)

  Que j'ai mis dans le projet(notebook)

- **Polars avec PostgreSQL et MinIO** (fichier `process_data.ipynb`)

Ces deux solutions présentent des avantages et des inconvénients selon le contexte d'utilisation. Cette analyse détaillée compare les deux méthodes en termes de performances, de flexibilité et d'adaptabilité aux grands volumes de données.

---

2. Présentation des deux solutions

2.1 PySpark sur Databricks

Avantages

- **Traitement massivement parallèle** : Adapté aux grandes volumétries de données.
- **Optimisation automatique** : Gestion efficace des ressources grâce à Spark.
- **Stockage distribué** : Compatible avec HDFS et Data Lakes.
- **Fonctionnalités avancées** : Support des DataFrames et RDDs.

Inconvénients

- **Complexité d'installation** : Nécessite un cluster Spark ou Databricks.
- **Moins adapté aux petites données** : Overhead important pour des jeux de données réduits.
- **Dépendance aux écosystèmes Big Data** : Doit être intégré à un environnement Spark.

2.2 Polars avec PostgreSQL et MinIO

Avantages

- **Rapidité sur des volumes moyens** : Polars est plus rapide que Pandas grâce à son moteur basé sur Rust.
- **Facilité de gestion des fichiers** : MinIO permet un stockage distribué comme S3.
- **Intégration avec PostgreSQL** : Permet d'effectuer des requêtes SQL optimisées.

- **Moins de ressources nécessaires** : Peut fonctionner sur une machine locale sans cluster.

Inconvénients

- **Moins efficace sur du Big Data** : Ne gère pas le calcul distribué comme Spark.
- **Configuration de MinIO** : Nécessite un serveur MinIO pour stocker les fichiers.
- **Pas de gestion automatique des tâches distribuées** : Doit être couplé à d'autres outils pour gérer de très grandes volumétries.

---

3. Comparaison détaillée

Critères	PySpark sur Databricks	Polars + PostgreSQL + MinIO
Performance	Très rapide sur Big Data (multi-nœuds)	Rapide sur données moyennes
Scalabilité	Adapté aux grands clusters distribués	Plus limité en mémoire RAM
Simplicité	Complexe à configurer, nécessite Spark	Facile à utiliser localement
Stockage	HDFS, Databricks, Cloud Storage	MinIO (équivalent Amazon S3)
Flexibilité	Doit s'intégrer dans un cluster	Peut fonctionner en local
Gestion SQL	SQL supporté via SparkSQL	Intégration native avec PostgreSQL

---

4. Comparaison détaillée entre Spark et Polars

4.1 Comportements

- **PySpark** repose sur un modèle de calcul distribué basé sur des RDDs et DataFrames. Il optimise automatiquement l'exécution grâce à son moteur Catalyst et Tungsten.
- **Polars**, basé sur Rust, adopte un modèle de traitement en mémoire optimisé pour le calcul vectoriel, permettant une manipulation rapide des données tabulaires.

4.2 Avantages et inconvénients

✅ Avantages de Spark

- Traitement massivement parallèle adapté aux grandes volumétries.
- Gestion automatique des ressources et optimisations internes.
- Compatible avec les écosystèmes Big Data (Hadoop, Databricks, AWS EMR).

❌ Inconvénients de Spark

- Nécessite un cluster et une configuration avancée.
- Moins performant pour des petits jeux de données en raison de son overhead.

✅ Avantages de Polars

- Très rapide sur des jeux de données moyens grâce à son architecture optimisée.
- Utilisation simple et exécution locale sans infrastructure complexe.
- Moins gourmand en mémoire comparé à Pandas et Spark pour certaines opérations.

❌ **Inconvénients de Polars**

- Ne gère pas nativement le calcul distribué sur plusieurs machines.
- Moins d'intégration avec les outils Big Data.

4.3 Performances

- **Spark** est plus performant pour traiter plusieurs téraoctets de données car il s'exécute sur un cluster distribué.
- **Polars** excelle sur des jeux de données allant de quelques centaines de milliers à quelques millions de lignes en exploitant au mieux la mémoire disponible.

---

5. Conclusion

Le choix entre **PySpark et Polars** dépend principalement du volume de données et des besoins en infrastructure :

- **Si les données sont massives (> 100M de lignes)**, PySpark est préférable grâce à son architecture distribuée.
- **Si les données sont moyennes (quelques millions de lignes)** et que l'on souhaite une exécution rapide sans infrastructure complexe, Polars est un excellent choix.
- **Si l'on veut un stockage structuré avec SQL**, PostgreSQL intégré avec Polars est plus efficace.

En conclusion, **PySpark est adapté aux projets Big Data**, tandis que **Polars est une alternative plus simple et plus rapide pour des volumes de données intermédiaires**.