Étude Comparative (pros et cons) sur les Formats de Fichiers : Parquet, ORC, Avro, Apache Arrow

1. Parquet

Pros:

- **Optimisé pour les requêtes analytiques** : Conçu pour être très efficace pour les lectures séquentielles et les requêtes analytiques complexes.
- **Colonnes compressées** : Utilise la compression par colonne, ce qui permet des taux de compression élevés.
- **Supporte les schémas complexes** : Peut gérer des types de données complexes tels que les tableaux et les structures imbriquées.
- **Interopérabilité** : Largement utilisé dans l'écosystème Hadoop et compatible avec de nombreux outils Big Data comme Apache Spark, Apache Hive, et Presto.

Cons:

- **Moins adapté pour les transactions** : Pas idéal pour les applications où des mises à jour fréquentes des données sont nécessaires.
- **Performance en écriture** : Les opérations d'écriture peuvent être moins performantes par rapport à d'autres formats comme Avro.

2. ORC (Optimized Row Columnar)

Pros:

- **Performances en lecture** : Optimisé pour les lectures rapides et les requêtes analytiques complexes, particulièrement performant pour les grands ensembles de données.
- **Compression efficace** : Offre une excellente compression grâce à des algorithmes comme Zlib.
- **Supporte les schémas complexes** : Peut gérer des structures de données complexes.
- **Optimisé pour Hive** : Particulièrement bien intégré avec Apache Hive.

Cons:

- **Complexité** : Peut être plus complexe à configurer et à optimiser par rapport à Parquet.
- **Compatibilité** : Moins de support natif en dehors de l'écosystème Hadoop, bien que cela s'améliore.

3. Avro

Pros:

- **Optimisé pour les opérations de lecture/écriture** : Très performant pour les opérations d'écriture et les lectures séquentielles.
- **Compatibilité des schémas** : Permet la gestion des versions des schémas, facilitant l'évolution des schémas de données.
- **Interopérabilité** : Large support dans l'écosystème Hadoop et au-delà, notamment avec Apache Kafka.
- **Compact** : Format binaire compact, ce qui réduit l'espace de stockage.

Cons:

- **Moins efficace pour les requêtes analytiques** : Pas aussi optimisé pour les requêtes analytiques complexes comme Parquet ou ORC.
- **Compression** : La compression est généralement moins efficace que Parquet ou ORC pour les requêtes analytiques.

4. Apache Arrow

Pros:

- **Vitesse de traitement** : Conçu pour des traitements de données en mémoire extrêmement rapides, facilitant l'interopérabilité entre différents systèmes.
- **Colonne orientée** : Utilise un format de colonne pour l'efficacité des analyses en mémoire.
- **Interopérabilité** : Permet l'échange de données efficace entre différents langages de programmation (Python, R, etc.) et frameworks (Pandas, Spark, etc.).
- **Zero-copy reads** : Permet des lectures de données sans copie, ce qui augmente la performance.

Cons:

- **En mémoire** : Principalement optimisé pour les opérations en mémoire, ce qui peut ne pas être idéal pour le stockage persistant.
- **Adoption** : Encore en cours d'adoption par rapport à des formats plus établis comme Parquet ou Avro.
- **Complexité** : Peut nécessiter une compréhension plus approfondie pour une intégration optimale.