Projet : Système de découpage intelligent de fichiers en Java

Détection de doublons et compression à la volée

Objectif général

Développer une application Java implémentant un **découpage intelligent de fichiers** basé sur l'algorithme **Content-Defined Chunking (CDC)**. Le système devra inclure une **détection des doublons** et une **compression en temps réel**, puis être validé par des **tests de performance** sur divers fichiers.

---

1. Fonctionnalités attendues

Phase 1 : Découpage dynamique des fichiers (Chunking)

- Implémentation de l'algorithme **Content-Defined Chunking (CDC)** pour découper les fichiers selon leur contenu.
- Utilisation de **Rabin Fingerprinting** pour identifier des points de coupure optimaux.
- Stockage des chunks en mémoire ou sur disque avec un index.

Phase 2 : Détection des doublons

- Calcul d'empreintes avec **SHA-1, SHA-256 ou BLAKE3**.
- Stockage des empreintes dans une **base de données indexée** (PostgreSQL, SQLite ou HashMap en mémoire).
- Vérification rapide pour éviter la duplication des blocs.

Phase 3 : Compression à la volée

- Application d'une **compression efficace** sur chaque chunk avec **Zstd, LZ4 ou Snappy**.
- Comparaison des performances entre **compression globale** et **compression par chunk**.

Phase 4 : Tests de performance

Métriques mesurées :

- Temps de découpage des fichiers.
- Gain de stockage grâce à la détection des doublons.
- Temps de reconstruction des fichiers.
- Impact de la compression sur la rapidité et l'efficacité.
- Tests sur différents types de fichiers (**texte, CSV, images, binaires, logs, archives ZIP**).

---

2. Technologies recommandées

Langage & Bibliothèques

- **Java 17+** pour bénéficier des dernières fonctionnalités.
- **Rabin Fingerprinting** : com.github.rabinfingerprint.rabin
- **Hashing** : java.security.MessageDigest (**SHA-256, BLAKE3**)
- **Compression** :
    - **LZ4** : net.jpountz.lz4
    - **Zstd** : com.github.luben:zstd-jni
    - **Snappy** : org.xerial.snappy
- **Base de données** : PostgreSQL ou SQLite (**JDBC avec HikariCP** pour la gestion de connexion).
- **Frameworks API (optionnel)** : **Spring Boot ou Quarkus** pour tester avec une API REST.