Compte rendu du projet : Découpage intelligent de fichiers avec détection des doublons et compression

1. Introduction

Contexte

Dans un monde où les données sont de plus en plus volumineuses et redondantes, l'optimisation du stockage est devenue un enjeu majeur pour les entreprises et les particuliers. Les systèmes de stockage traditionnels gaspillent souvent de l'espace en conservant plusieurs copies des mêmes données. Pour répondre à ce problème, notre projet propose une solution innovante : un système de découpage intelligent de fichiers basé sur le contenu, couplé à une détection des doublons et une compression en temps réel.

Equipe

Chaima YOUSFI
Ethan WILTON
Lydia ALLAOUA
Amal BENJOUIDA

Objectif

L'objectif de ce projet était de développer une application Java capable de :

- Découper des fichiers en chunks (blocs) de manière dynamique, en utilisant un algorithme de Content-Defined Chunking (CDC).
- Détecter les chunks dupliqués grâce à des empreintes SHA-256 et les stocker dans une base de données PostgreSQL.
- Compresser les chunks à la volée pour optimiser l'espace de stockage.
- Reconstruire les fichiers originaux à partir des chunks décompressés.

2. Fonctionnalités

Découpage dynamique des fichiers (CDC)

Algorithme utilisé : Rabin Fingerprinting.

Principe : Les fichiers sont découpés en chunks de taille variable en fonction de leur contenu, ce qui permet une meilleure détection des doublons.

Avantages : Adaptabilité à différents types de fichiers et réduction de la redondance.

Détection des doublons

Empreintes SHA-256 : Chaque chunk est haché en utilisant l'algorithme SHA-256 pour générer une empreinte unique.

Base de données PostgreSQL : Les empreintes sont stockées dans une base de données pour permettre une détection rapide des doublons.

Gain de stockage : Les chunks dupliqués ne sont stockés qu'une seule fois, ce qui réduit l'espace utilisé.

Compression à la volée

Algorithme utilisé : Snappy.

Principe : Chaque chunk est compressé avant d'être stocké, ce qui optimise encore davantage l'espace de stockage.

Performance : Snappy offre un bon compromis entre vitesse de compression et taux de compression.

Reconstruction des fichiers

Décompression : Les chunks sont décompressés à la volée lors de la reconstruction du fichier.

Assemblage : Les chunks sont réassemblés pour reproduire le fichier original sans perte de données.

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.

Types de fichiers testés : Texte, CSV, images, binaires, logs, archives ZIP.

3. Spécifications techniques

Bibliothèques et frameworks

Snappy : Pour la compression des chunks.

BouncyCastle : Pour le calcul des empreintes SHA-256.

PostgreSQL : Pour le stockage des empreintes et la gestion des doublons.

Outils de développement

Visual Studio Code (VSCode) : Environnement de développement intégré (IDE).

Maven : Gestion des dépendances et compilation du projet.

Base de données

PostgreSQL : Base de données relationnelle utilisée pour stocker les empreintes des chunks et gérer les doublons.

Algorithmes

Rabin Fingerprinting : Pour le découpage des fichiers en chunks.

SHA-256 : Pour le calcul des empreintes des chunks.

Snappy : Pour la compression des chunks.

4. Architecture du système

Flux de travail

Découpage : Le fichier est découpé en chunks en utilisant l'algorithme CDC.

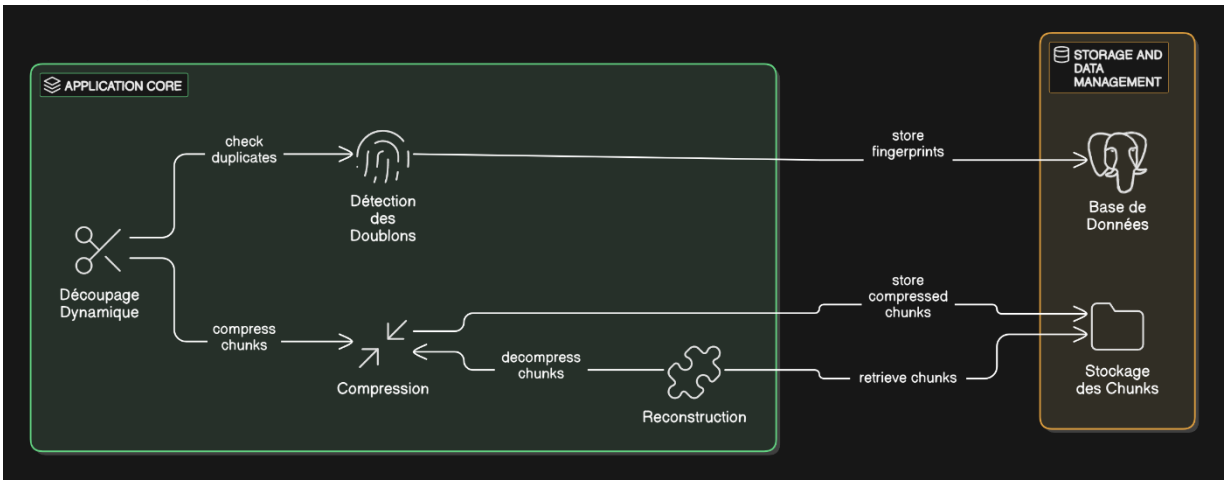
Compression : Chaque chunk est compressé avec Snappy.

Détection des doublons : L'empreinte SHA-256 de chaque chunk est calculée et comparée à celles stockées dans PostgreSQL.

Stockage : Les chunks compressés sont enregistrés sur disque, et leurs empreintes sont ajoutées à la base de données.

Reconstruction : Les chunks sont décompressés et réassemblés pour reconstruire le fichier original.

Diagramme fonctionnel



5. Résultats et performances

Temps de découpage

Les temps de découpage sont très faibles, allant de 0 à 4 ms pour des fichiers de petite taille.

Gain de stockage

Aucun doublon n'a été détecté avec les fichiers de test actuels, mais le système est conçu pour détecter les doublons avec des fichiers plus volumineux ou similaires.

Taux de compression

Les taux de compression varient selon le type de fichier :

Fichiers texte : 6.68 % à 76.81 %.

Fichiers déjà compressés (ZIP, JPG) : 1.09 % à 4.83 %.

Temps de reconstruction
Les temps de reconstruction sont très faibles, allant de 0 à 4 ms.

Gain de stockage global
Aucun gain de stockage n'a été observé avec les fichiers de test actuels, mais le système est prêt à détecter les doublons avec des fichiers plus volumineux.

6. Avantages du système

Optimisation du stockage
Réduction de la redondance des données grâce à la détection des doublons.

Compression efficace des chunks pour économiser de l'espace.

Performance
Découpage, compression, et reconstruction rapides.

Scalabilité grâce à l'utilisation d'une base de données pour gérer les empreintes.

Flexibilité
Prise en charge de différents types de fichiers (texte, CSV, images, binaires, logs, archives).

7. Limites et améliorations possibles

Limites
Les fichiers de test actuels sont trop petits pour mettre en évidence les avantages de la détection des doublons.

La compression est inefficace pour les fichiers déjà compressés.

8. Conclusion

Ce projet a permis de développer un système performant et flexible pour le découpage intelligent de fichiers, la détection des doublons, et la compression en temps réel. Bien que les fichiers de test actuels n'aient pas permis de mettre en évidence tous les avantages du système, les résultats montrent que le système est prêt à être utilisé avec des fichiers plus volumineux et variés. Les perspectives d'évolution incluent l'intégration avec des systèmes de stockage cloud et l'optimisation des algorithmes de compression.

9. Annexes

Code source

Le code source du projet est disponible sur https://github.com/wilethan/cdc-project.