République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université SAAD DAHLEB- BLIDA 1

Faculté des sciences

Département d'informatique

Mémoire de fin d'études pour l'obtention d'un diplôme de Master en informatique.

Option : Systèmes Informatique et Réseau (SIR)

Thème

Un Algorithme de Compression d'Image Distribué Basé sur le Système HADOOP

Réalisé par :

- Manel GHOUALI
- Meriem BOUTAHRAOUI

Soutenue le 09/07/2023 Devant le jury composé de :

-Mr. M.BALA	Président
-Mme HADJ HENNI	Examinatrice
-Mme.K. Midoun	Promotrice

Résumé

La compression d'image est une technique visant à réduire la taille d'une image, et elle peut être réalisée de manière avec ou sans perte de données. Dans cette étude, nous avons mis en œuvre deux approches de compression, à la fois séquentielle et parallèle utilisant Hadoop MapReduce, en utilisant les algorithmes RLE et Huffman. Une comparaison approfondie des performances des deux approches a été réalisée en évaluant leur taux de compression et leur temps d'exécution. L'objectif était de déterminer quelle approche offrir les meilleurs résultats en termes de compression et d'efficacité temporelle.

MOTS-CLÉS : MapReduce, compression d'image ,Hadoop .

ملخص

ضغط الصور هو أسلوب لتقليل حجم الصورة ، ويمكن القيام به بطريقة ضياع أو بدون فقدان. في هذه الدراسة ، قمنا بتنفيذ نهجين للضغط ، متسلسل ومتوازي باستخدام Hadoop MapReduce ، باستخدام خوارزميات RLE و Huffman. تم إجراء مقارنة متعمقة لأداء كلا المنهجين من خلال تقييم معدل الضغط ووقت التنفيذ. كان الهدف هو تحديد النهج الذي يقدم أفضل النتائج من حيث الضغط وكفاءة الوقت.

الكلمات المفتاحية: خوارزمية ضغط الصور , الأداء, MapReduce

Abstract

Image compression is a technique to reduce the size of an image, and it can be done in a lossy or lossless manner. In this study, we implement two image compression algorithms :RLE and Huffman using Hadoop/MapReduce. The evaluation of the performance of these approaches is based on their compression ratio and execution time. The objective was to determine which approach offered the best results in terms of compression and time efficiency.

KEYWORDS : MapReduce, Image compression,HADOOP.

Remerciement

Je Remercier mon Dieu de m'avoir donné la patience et le courage pour réaliser ce travail.

Je tiens à remercier tout particulièrement notre promotrice Madame

Midoun Khadidja pour avoir proposé un sujet d'actualité aussi intéressant ainsi que pour ses conseils tout au long de ce travail.

Je tiens à remercier Monsieur M. BALA, pour avoir l'honneur de présider le jury.

Je tiens à remercier Madame HADJ HENNI, pour l'honneur qu'elle me fait participer au jury.

Je tiens à remercier ici toutes les personnes : enseignants, étudiants avec qui j'ai eu le plaisir de travailler de manière directe ou indirecte durant ces mois. Qu'ils y trouvent une expression de ma reconnaissance.

Dédicaces

A mes très chers parents, qui sont toujours à mes côtés que Dieu

Vous accorde une longue et heureuse vie.

A mes frères et ma sœur.

A toute ma famille.

A tous mes amies, qu'ils m'excusent de ne pas pouvoir les citer au risque d'oublier

Quelqu'un.

A tous ceux qui ont contribué un jour à notre éducation et formation.

A tous je dédie ce travail.

Manel

Dédicaces

Tout d'abord, je tiens à remercier DIEU De m'avoir donné la force et le courage de mener à bien ce modeste travail. Je tiens à dédier cet humble travail à :

A ma tendre mère et mon très cher père Leurs prières et leurs conseils m'ont toujours accompagné. Je les remercié pour tous ce que m'ont donné (Confiance, moyens et amour).

A mes frères : Zaki et Yacine.

A ma sœur : Amina

Ma chère grand-mère Mani « zhor » qui nous a quitté très tôt.

A mon binôme : manel

Tous ceux qui m'aiment et que j'aime

Meriem

Sommaire

Table des Figures

Liste des tableaux

Liste des abréviations

RLE: Run-Length Encoding

LZW: Lempel-Ziv-Welch

HDFS: Hadoop Distributed File System

Hadoop: High-Availability Distributed Object-Oriented Platform

INTRODUCTION GENERALE

Introduction Générale

Avec la forte augmentation de l'utilisation des médias sociaux parmi les gens, la quantité d'images et de vidéos transmises augmente considérablement [1], [2].Par exemple, en janvier 2019, plus de 240 milliards de photos ont été téléchargées par les utilisateurs du site Facebook, avec 350 millions de nouvelles photos quotidiennes. Cela exige un montant énorme de stockage et d'une liaison de communication très rapide [3].La compression d'image est une discipline clé dans le domaine du traitement des images et de la gestion des données visuelles [4]-[5].Les images numériques occupent souvent un espace considérable sur les supports de stockage et peuvent nécessiter une transmission à haut débit. La compression d'image vise à réduire la taille des fichiers image tout en préservant une qualité visuelle acceptable. Elle joue un rôle crucial dans de nombreux domaines, tels que la transmission d'images sur des réseaux à bande passante limitée, le stockage d'images dans des dispositifs de mémoire restreinte et le traitement rapide des images dans des applications interactives.Il existe deux catégories principales de techniques de compression d'image : sans perte et avec perte. En compression instantanée sans perte, la qualité de l'image compressée est similaire à celle de l'image originale. Toutes les informations de l'image originale sont conservées dans l'image compressée [6].D'autre part, la compression d'image avec perte modifie légèrement les informations de l'image originale d'une manière qui diminue la qualité de l'image compressée par rapport à l'image d'origine [7]-[8].

La croissance exponentielle du volume de données visuelles, telles que les images numériques, représente un défi majeur en termes de stockage, de transmission et de traitement. La compression d'image offre une solution efficace pour réduire la taille des fichiers image sans compromettre de manière significative leur qualité visuelle. L'utilisation de modèles de programmation parallèle tels que Hadoop MapReduce permet d'exploiter le potentiel du calcul parallèle pour accélérer le processus de compression d'image sur des clusters de machines.

L'utilisation de modèles de programmation parallèle tels que Hadoop MapReduce permet d'exploiter le potentiel du calcul parallèle pour accélérer le processus de compression d'image sur des clusters de machines.

Ce projet de fin d'études vise à explorer et à évaluer les performances et l'efficacité de l'implémentation parallèle distribué des algorithmes de compression d'image (RLE et Huffman) en utilisant le framwork Hadoop/MapReduce. L'objectif est de fournir une compréhension approfondie des avantages et des limitations de l'utilisation de Hadoop/MapReduce pour la compression d'image, ainsi que des recommandations pour son application dans des scénarios réels

Ce mémoire est organisé en trois chapitres comme suit :

- Dans le premier chapitre, nous allons présenter les différents concepts liés aux compressions d'image, ensuite nous allons aborder les types et les méthodes des prétraitements de la compression d'images.

- Le deuxième chapitre est consacré pour la conception de notre solution.
- Enfin dans le troisième chapitre, nous allons exposer les différents tests pratiques réalisés avec une évaluation des résultats trouvés.

CHAPITRE 1 : LA COMPRESSION D'IMAGES

1. **Introduction**

La croissance rapide des médias sociaux et des réseaux numériques s'est traduite par l'accès et l'échange quotidiens d'énormes quantités de données d'images. Cependant, une image non optimisée peut avoir une taille énorme. La compression d'image est un processus de réduction de la taille d'un fichier image tout en préservant le maximum de détails possible. Elle est utilisée dans de nombreux domaines tels que : les réseaux sociaux, médecine et jeux vidéo pour améliorer l'efficacité de stockage, de traitement et de transmission selon le domaine d'utilisation. Ce chapitre présente les notions de base d'images, les types de compression d'image, le framework Hadoop/MapREduce avec son architecture .

2. Les images numériques

L'image est une représentation d'une personne ou d'un objet, peut être obtenue soit à partir des capteurs optiques (caméra, scanneur…) soit créée à l'aide d'un logiciel. [9].Pour faire le traitement d'une image avec les outils informatiques, il faut qu'elle soit numérisée. L'image numérique est l'image dont la surface est divisée en éléments de taille fixe appelés pixel. Elle est représentée par une matrice bidimensionnelle où chaque élément de la matrice correspondant à un pixel de l'image[10]., comme il est illustré dans la figure ci-dessous (Figure 1).

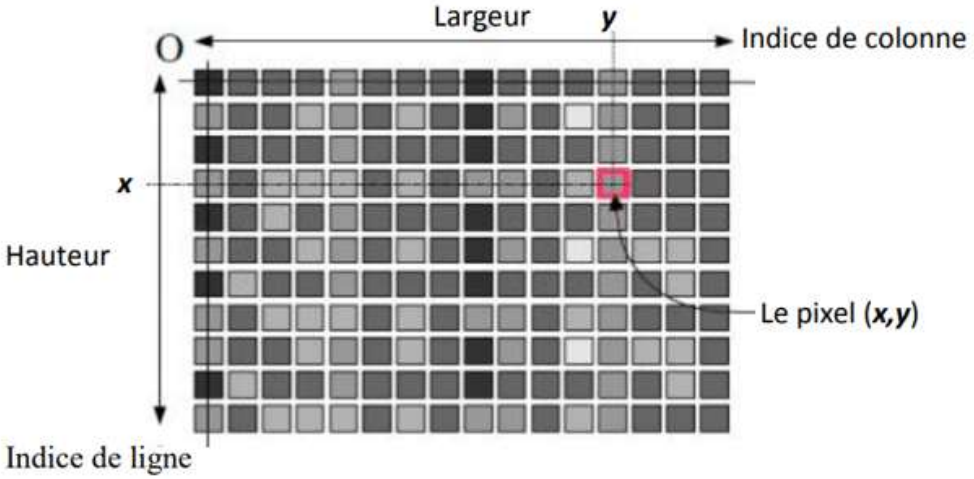

Figure 1: Image Numérique

2.1. Les caractéristiques des images numériques

L'image est un ensemble structuré d'informations parmi ses caractéristiques nous pouvons citer les paramètres suivants :

1) Le codage de la couleur

On distingue généralement trois grandes représentations de couleurs pour une image numérique : le noir et blanc (figure 2.a)où chaque pixel ne peut prendre que deux valeurs distinctes pour représenter les deux couleurs le noir et le blanc, le niveau de gris (figure 2.b) où chaque pixel est représenté par une valeur allant de 0 à 255[11]. et le cogade de couleur (figure 2.c) qui permet de représenter des images en couleur e générale e utilisant le modèle RVB (rouge, vert, bleu), où chaque pixel est composé de trois valeurs numériques correspondant à l'intensité de chaque couleur. Ces valeurs peuvent varier de 0 à 255 pour chaque couleur, ce qui permet une large gamme de couleurs[12].

(a *(b)* *(c)*

Figure 2 : Les différents codages

2) Le pixel

C'est l'unité de base permettant de mesurer la définition d'une image. Chaque pixel d'une image est caractérisé par sa localisation dans le repère de l'image (les deux coordonnées x et y) et sa couleur [10].

3) Voisinage d'un pixel

C'est l'ensemble des pixels qui sont situés autour de ce pixel dans une image. Le voisinage est utilisé lors du traitement d'une image. On distingue deux types de voisinage : (1) Voisinage à 4-voisins (figure 3. a) : On ne prend en considération que les pixels qui ont un côté commun avec le pixel considéré. (2) Voisinage à 8-voisins (figure 3.b) : On prend en compte tous les pixels qui ont au moins un point en liaison avec le pixel considéré.

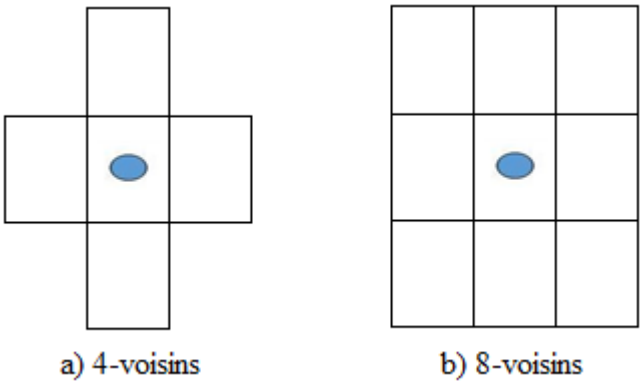

Figure 3 : Le voisinage d'un pixel

4) **La définition**

C'est le nombre de pixel que comporte une image numérique en largeur et en hauteur [13].

5) La résolution

La résolution est le nombre de pixels par unité de longueur. est exprimée le plus souvent en ppp (point par pouces) ou en dpi (dots per inch), avec: 1 pouce = 2.54 cm.

La résolution définit la netteté d'une image et sa qualité d'affichage à l'écran. Plus la résolution est grande (c'est-à-dire plus il y a de pixels dans une longueur de 1 pouce), plus votre image est précise dans les détails [13].

$$\text{Résolution} = \frac{\text{définition}}{\text{dimension}}$$

6)Contour

Les contours [14] représentent la frontière entre les objets de l'image, ou la limite entre deux pixels dont les niveaux de gris représentent une différence significative. Les textures décrivent la structure de ceux-ci. L'extraction de contour consiste à identifier dans l'image les points qui séparent deux textures différentes[8].

7)Luminance

C'est le degré de luminosité des pixels de l'image. Elle est substituée au mot brillance, qui correspond à l'éclat d'un objet. Il est conseillé d'éviter les images où la gamme de contraste tend vers le blanc ou le noir, ces images entraînent des pertes de détails dans les zones sombres ou lumineuses [11].

8)**Histogramme**

L'histogramme est un vecteur H, où chaque élément de ce vecteur, H(i), représentesa fréquence d'apparition d'un pixel i. On peut donc assimiler l'histogramme à la densité de probabilité des intensités lumineuses. Puisque le nombre des pixels est généralement assez grand, on peut alors normaliser l'effectif de chaque niveau de gris en divisant chaque terme du tableau par la surface du plan exprimée en nombre total des pixels de l'image. De ce fait, chaque case H(i) du vecteur représente la probabilité d'avoir l'intensité « i ».

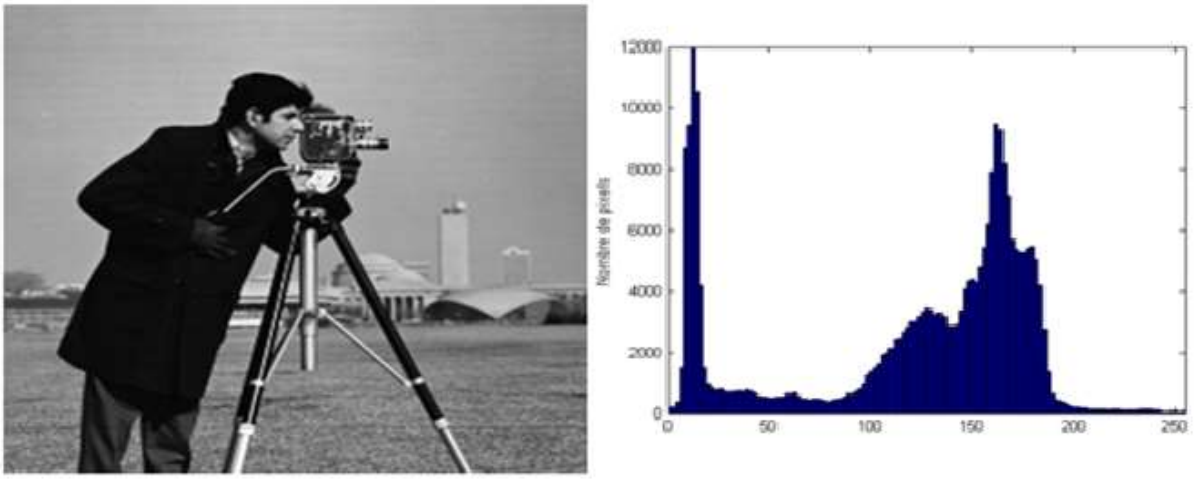

Figure 4: Un exemple d'un histogramme d'une image

3. La compression d'image

La compression d'image est un processus qui vise à créer une version compressée d'une image. En réduisant la quantité de données requises pour représenter une image, ce qui permet de diminuer les besoins en stockage et en transmission. Pour compresser une image, on élimine ou réduit trois types de redondances de données : la redondance de codage, l'inter-redondance et la redondance psycho-visuelle.

La redondance de codage se produit lorsque des mots de code optimaux ne sont pas utilisés de manière efficace. L'inter-redondance découle des corrélations entre les pixels de l'image numérique. Enfin, la redondance psycho-visuelle concerne les données que le système visuel humain ignore, car elles représentent des informations visuelles non essentielles[15].

Les domaines d'utilisation de la compression d'image sont nombreux et variés. Voici quelques exemples :

- Applications médicales : La compression d'image est essentielle dans les domaines de la radiologie et de l'imagerie médicale. Elle permet de stocker et de transmettre des images médicales, telles que des radiographies, des IRM (Imagerie par Résonance Magnétique) ou des scanners, de manière plus efficace, tout en préservant les détails nécessaires pour un diagnostic précis Les algorithmes de compression sans perte sont plus adaptés à ce type d'images car aucune perte d'information ne peut être tolérée[16].

- Vidéo surveillance : Dans le domaine de la sécurité et de la surveillance, les systèmes de vidéosurveillance génèrent de grandes quantités de données vidéo. La compression d'image est utilisée pour réduire la taille des vidéos et faciliter leur stockage et leur analyse ultérieure.
- Transmission et stockage d'images sur Internet : La compression d'image permet de réduire la taille des fichiers images, ce qui facilite leur transfert rapide à travers les réseaux et leur stockage sur les dispositifs de stockage limités en espace. Cela est particulièrement important pour les applications Web, les réseaux sociaux, les services de messagerie, etc

l'objectif de la compression d'image est de réduire la taille du fichier image tout en préservant une qualité visuelle acceptable, ce qui permet de :

- Réduire les besoins en espace de stockage pour les images.
- Faciliter le transfert et la transmission rapide des images sur les réseaux.
- Optimiser l'utilisation de la bande passante lors de la diffusion en continu d'images ou de vidéos.
- Permettre le stockage et la transmission efficace d'images dans des domaines tels que la médecine, la surveillance, la télévision, etc.
- Améliorer l'expérience utilisateur en réduisant les temps de chargement et en permettant une visualisation fluide des images[17].

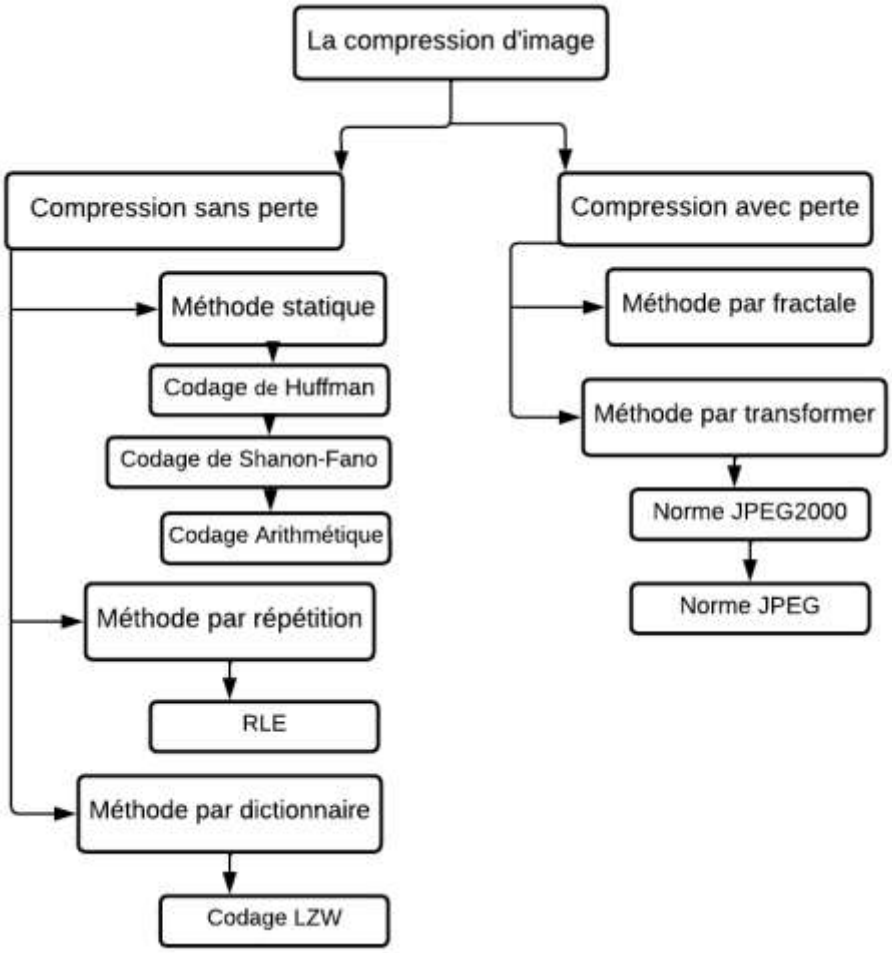

Figure 5 : Méthodes de compression d'image

3.1. Les types de la compression d'image

3.1.1 Compression avec perte

La compression avec perte de données est couramment utilisée avec succès pour certains types de fichiers, tels que l'audio, les vidéos et les images, ce qui permet d'obtenir un haut niveau de compression et de bons résultats. Elle réduit la taille du fichier en éliminant définitivement les informations redondantes. On distingue deux méthodes pour ce type de compression, La méthode fractale et la méthode de la transformation. [18]

Méthode par fractale

Elle consiste à mettre en relation un modèle mathématique avec celle-ci afin de lui appliquer un transformation contractante. Il faudra ensuite définir les auto-similarités, afin de ne les sauvegarder qu'une seule fois[19].

Méthode par transformer

Elle vise à réduire les redondances fréquentielles en transformant les pixels d'un espace fortement corrélé vers un espace moins corrélé[20]. Cette transformation est suivie d'une quantification et d'un codage entropique dans divers algorithmes de compression d'image tels que JPEG (utilisant la transformation DCT)[21]. et JPEG2000 (utilisant la transformation en ondelettes DWT)[22].

3.1.2. Compression sans perte

La compression sans perte supprime également des données, mais elle peut restaurer l'original si nécessaire. L'objectif est de conserver une qualité élevée, tout en réduisant la taille du fichier, La compression sans perte fait ce qu'elle indique dans son nom : Elle comprime la taille du fichier d'une image autant que possible sans affecter la qualité visible. Pour cela, elle supprime les métadonnées de l'image, qui peuvent occuper un espace inutile[23].On distingue trois grades méthodes pour ce type de compression, La méthode statique, la méthode par répétition et la méthode par dictionnaire.

Méthode statique :

- **Codage d'Huffman**

Huffman, dans son ouvrage [Huf 52], a proposé une méthode statistique pour attribuer des mots de code binaires aux différents pixels à compresser. La méthode prend en compte la probabilité d'occurrence de chaque pixel de l'image et assigne des codes courts aux symboles les plus fréquents et des codes longs aux symboles les plus rares. En utilisant cette approche, la séquence finale de pixels codés avec des longueurs variables sera plus petite que la taille originale de l'image [24].

Tout d'abord, l'image est analysée afin de déterminer les fréquences d'apparition de chaque symbole. Dans le contexte de la compression d'image, les symboles peuvent représenter les valeurs des pixels, des combinaisons de pixels ou d'autres éléments de l'image. Le codage Huffman crée un arbre ordonné à partir de tous les symboles et de leurs fréquences d'apparition. Les branches de l'arbre sont construites de manière récursive, en commençant par les symboles les plus fréquents. Le code de chaque symbole correspond à la séquence des codes le long du chemin allant du symbole à la racine de l'arbre. Plus le symbole est profond dans l'arbre, plus la quantité de bits nécessaire pour le représenter est importante.la figure suivante montre un exemple de codage d'Huffman.

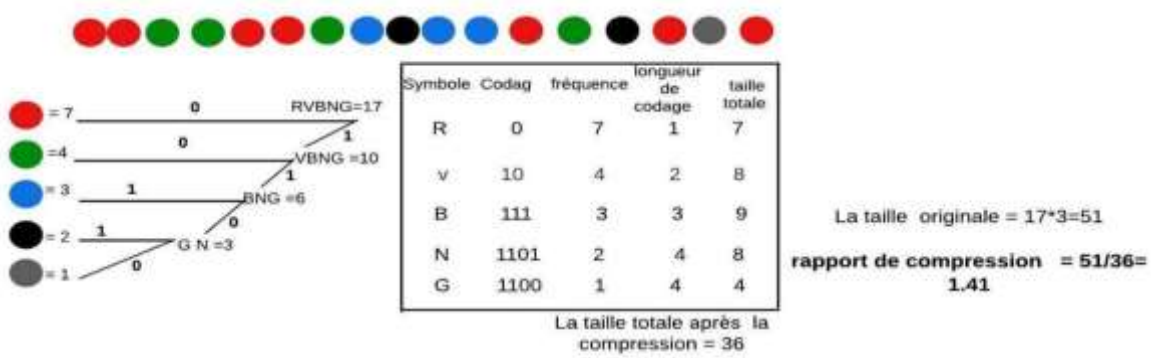

Figure 6 : Algorithme de compression sans perte de Huffman

- **Codage de Shanon-Fano**

C'est le même principe de Huffman mais dans la 2ème étape,au lieu de construire l'arbre de Huffman, on construit le tableau de Shano-Fano. ([25 ,26,27])

- **Codage Arithmétique**

Le codage arithmétique (CA) [28] est un codage statistique qui attribue à une suite de symboles une valeur réelle. Il consiste à découper l'intervalle des réels [0, 1[en sous-intervalles, dont les longueurs sont fonction des probabilités des symboles. Le codage arithmétique n'attribue pas un code à chaque symbole comme Huffman et les autres codages par blocs, mais un code au message tout entier

Méthode par répétition

- **Codage RLE**

L'algorithme RLE (Run-Length Encoding) est une méthode simple et efficace de compression d'image. Il repose sur le principe de l'encodage par longueur de séquence. Dans le contexte de la compression d'image, l'algorithme RLE identifie les séquences consécutives de pixels ayant la même valeur et les remplace par une paire composée de la valeur du pixel et du nombre de fois où cette valeur se répète. Par exemple, une séquence de pixels noirs consécutifs serait représentée par "0, n" où 0 est la valeur du pixel noir et n est le nombre de fois qu'il se répète. Cette compression permet de réduire la taille du fichier en éliminant les répétitions de pixels et en les représentant de manière plus concise.

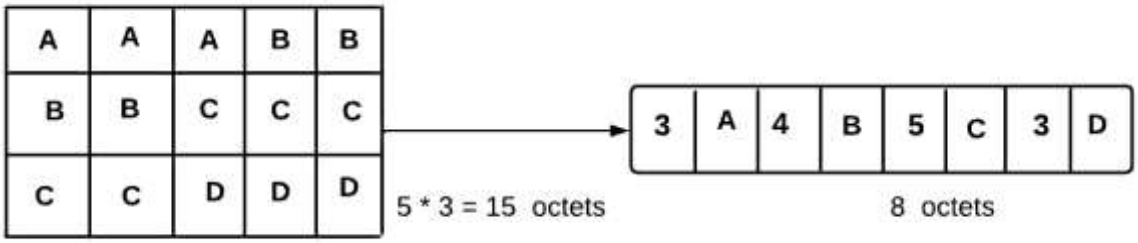

Figure 7 : Principe de l'algorithme RLE

Méthode par dictionnaire

- **Codage LZW**

L'algorithme LZW (Lempel-Ziv-Welch) est un algorithme de compression sans perte. Cet algorithme est le fruit d'un travail collectif des trois professeurs : Abraham Lampel, Jacob Ziv et Terry Welch[---]. II a pour but de minimiser la taille des mots qui se répètent fréquemment dans un message transmis. II est une méthode de type dictionnaire car au cours de la compression de données, il utilise les dictionnaires[29].

Dans le tableau ci-dessous, nous récapitulons les principaux points de divergence et de similarité entre les techniques de compression avec perte et sans perte[30] [31].

Type de compression	La compression sans perte	La compression avec perte
le principe	C'est un algorithmes de compression de données qui permet de reconstruire avec précision les données d'origine à partir des données compressées.	C' est une méthode de codage qui utilise des estimations imprécises pour représenter le contenu.
utilisé dans	Texte ou programme, images et son.	Images, audio et vidéo.

Application	RAW, BMP, PNG, WAV, FLAC, ALAC,..., etc.	JPEG, GUI, MP3, MP4, OGG, H-264, MKV,..... ,etc.
Capacité de garder le contenu stockage de l'image	Grande	Fiable

Tableau 1.1 : Comparatif des types de compression

Les deux algorithmes RLE et Huffman sont appliqués dans ce projet permettent de reconstituer les valeurs exactes des pixels de l'image originale. La partie suivante inclut le principe de fonctionnement de chacun.

3.2. Évaluation des facteurs clés pour la compression d'images

Lors de la compression d'images, il est essentiel d'évaluer plusieurs facteurs clés dans la compression pour mesurer la performance d'un algorithme.

Le rapport de compression d'image est une mesure qui évalue l'efficacité d'un algorithme de compression pour une image donnée. Il est calculé en comparant la taille de l'image d'origine avec la taille de l'image compressée. [32], [33].

$$Rapport = CR = \frac{\textbf{nombre de bits de l'image avant compression}}{\textbf{nombre de bits de l image après compression}} = \frac{\boldsymbol{R_0}}{\mathbf{R}}$$

Parallèlement, le taux de compression exprime en pourcentage l'espace obtenu après la compression par rapport à l'espace total requis par les données avant la compression. Il est calculé à l'aide de la formule

$$\mathbf{T}=\frac{\mathbf{1}}{\mathbf{1-rapport\ de\ compression}}*100$$

En outre, le temps de traitement est un facteur essentiel à considérer lors de l'évaluation des performances de la méthode de compression. Il mesure le temps nécessaire pour effectuer les opérations de compression et de décompression des images. La contrainte temporelle peut varier en fonction de l'application visée par la compression .

Il est donc crucial de prendre en compte ces facteurs pour garantir une compression d'image efficace, tout en répondant aux exigences spécifiques de l'application concernée.

4. Problématique

Les images sont généralement volumineuses en termes de taille de fichier, ce qui peut poser des défis en termes de transfert, de stockage et de traitement. La compression d'image vise à réduire la taille des fichiers d'image tout en minimisant la perte de qualité perceptible. Les images de haute résolution peuvent occuper un espace de stockage considérable, ce qui pose des défis en termes de capacité de stockage. De plus, la transmission d'images sur des réseaux à bande passante limitée peut entraîner des retards, une dégradation de la qualité et des problèmes de transfert.

La compression d'image joue un rôle essentiel dans la résolution de ces problèmes. En réduisant la taille des images, la compression permet d'économiser de l'espace de stockage et de réduire la bande passante nécessaire pour la transmission. Cependant, il existe un compromis entre la taille du fichier compressé et la qualité visuelle de l'image. Par conséquent. De plus, le traitement séquentiel traditionnel peut prendre beaucoup de temps pour compresser et décompresser les images, ce qui peut entraîner des retards indésirables dans les applications nécessitant une compression en temps réel.

La solution proposée consiste à utiliser le traitement parallèle avec le Framework Hadoop. Hadoop est un système de traitement distribué qui permet de diviser les tâches en plusieurs sous-tâches et de les exécuter en parallèle sur un cluster de machines. En utilisant Hadoop MapReduce, les opérations de compression d'image peuvent être réparties sur plusieurs nœuds du cluster, ce qui permet de traiter simultanément plusieurs parties de l'image. Cela permet d'accélérer considérablement le processus de compression et de décompression des images.

La solution proposée dans notre projet est la compression d'image avec le traitement parallèle Hadoop MapReduce visant à résoudre les problèmes de stockage et de transmission d'images, ainsi qu'à réduire le temps de traitement séquentiel.

5. Hadoop

Hadoop(High-availability distributed object-oriented platform) est un Framework open source pour le traitement, le stockage et l'analyse de grandes quantités de données distribuées [34]. qui est similaire au système de fichiers principal de Google. En tant qu'environnement logiciel de code source libre conçu pour le traitement et l'analyse de grands volumes de données de manière distribuée, Hadoop peut être utilisé dans la compression d'image pour effectuer des opérations de compression et de décompression en parallèle sur un cluster de machines. Cela permet d'exploiter efficacement les capacités de calcul parallèle et de réduire les temps de traitement par rapport aux approches séquentielles traditionnelles [35]. Hadoop C’est la partie centrale de l’infrastructure informatique pour beaucoup d' entreprises comme Facebook [36], LinkedIn [37], Twitter [38] et le New York Times [39].

5.1. Architecture Hadoop

Hadoop repose en deux grande partie le stockage de données HDFS (Hadoop Distributed File System) et le modèle de traitement parallèle Map Reduce.

5.1.1 HDFS

HDFS (Hadoop Distributed File System) est un système de stockage distribué qui permet le traitement parallèle et distribué des données, ainsi que la tolérance aux erreurs grâce à la duplication des données. Il est utilisé comme principal système de stockage de données par les applications Hadoop. HDFS utilise une architecture avec un "NameNode" (noeud maître) qui gère l'espace de noms du système de fichiers et régule l'accès aux fichiers, et des "DataNodes" (noeuds esclaves) qui gèrent le stockage attaché aux nœuds[40]. Dans le contexte de la compression d'image, HDFS est utilisé pour stocker et répartir les fichiers d'images sur plusieurs nœuds d'un cluster, offrant une haute disponibilité, une tolérance aux pannes et une capacité de traitement parallèle. Il divise les fichiers d'images en blocs de données et les distribue sur différents nœuds, permettant ainsi une lecture et une écriture simultanées des données. En utilisant HDFS dans la compression d'image, il est possible de bénéficier d'une gestion efficace des ressources, d'une répartition de charge optimale et d'une réduction significative des temps de traitement. HDFS est un élément clé de l'écosystème Hadoop, fournissant un moyen fiable de gérer des pools de données volumineuses et de prendre en charge les applications d'analyse de données volumineuses.

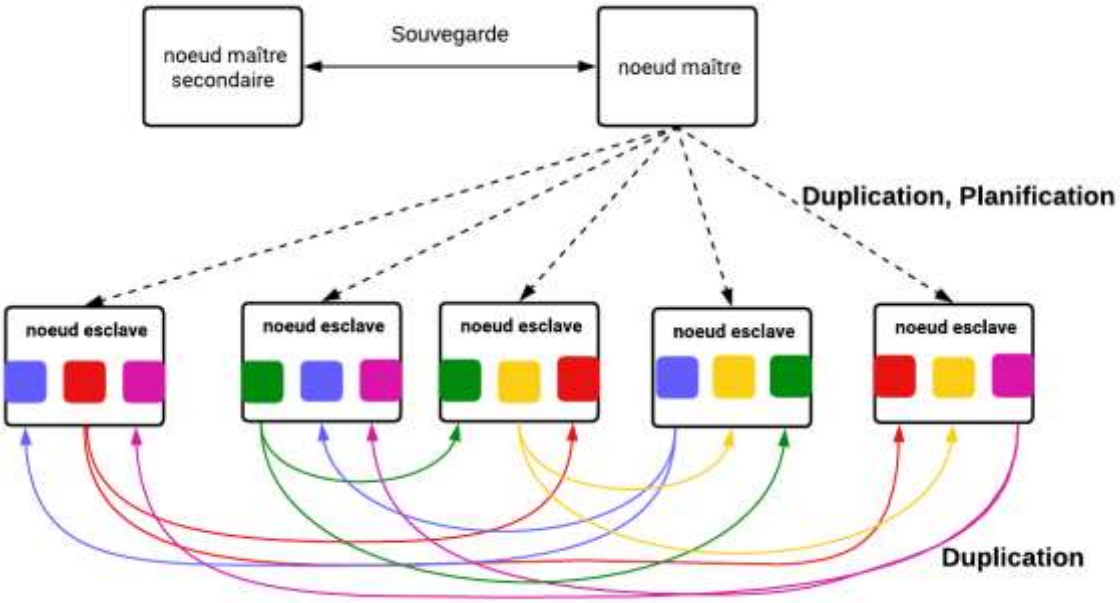

Figure 8 : Schéma de principe du HDFS

5.1.2 MapReduce

MapReduce est un modèle de programmation qui permet de calculer en parallèles de manière distribuée sur des données très volumineuses Il permet une répartition efficace du traitement des données en les divisant en sous-problèmes et en les traitant parallèlement sur un cluster de nœuds [43][41].MapReduce permet d'exploiter efficacement les ressources disponibles, d'optimiser les performances et de réduire les temps de traitement par rapport aux approches séquentielles traditionnelles. [42]

Le modèle de programmation MapReduce repose sur deux fonctions essentielles : map() et reduce().

- Dans la phase de Map, chaque nœud analyse un problème, le divise en sous-problèmes, puis les délègue à d'autres nœuds qui peuvent également diviser les sous-problèmes de manière récursive. Les différents nœuds traitent ensuite les sous-problèmes en utilisant la fonction Map, qui associe un couple clé-valeur à un ensemble de nouveaux couples clé-valeur :

map(clé1, valeur1) → liste(clé2, valeur2)

Ensuite, nous abordons la phase Réduire, qui intervient après que les nœuds inférieurs ont transmis leurs résultats au nœud parent qui les avait sollicités. À ce stade, le nœud parent effectue un calcul partiel en utilisant la fonction Réduire (réduction) pour associer toutes les valeurs correspondantes à la même clé à une unique paire (clé, valeur). Cette paire résultante est ensuite transmise au nœud supérieur.

Au terme de ce processus, le nœud d'origine est en mesure de reconstituer une réponse au problème initial qui lui avait été soumis. Cela se traduit généralement par une transformation de la forme des données, comme illustré ci-dessous :

reduce(key2,list(valeur2))→ (key2, valeur3)

Le modèle MapReduce se distingue par sa capacité à répartir efficacement le traitement des données en les partageant en sous-problèmes, puis en les traitant de manière parallèle sur un cluster de nœuds. Cette approche permet de tirer pleinement partie des ressources disponibles pour accélérer le traitement global[43] .

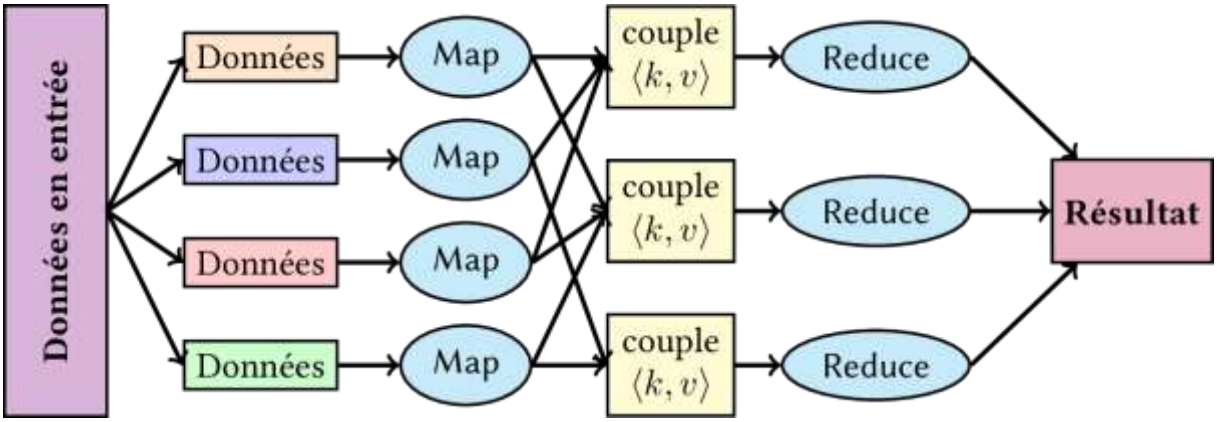

Figure 9 : Schéma de fonctionnement du MapReduce

6. Travaux Connexe :

ARTICLE	Pert problem		MapReduce phase		Type image	Idée clé	avantage	limitation
	Sans pert	Avec pert	MAP	REDUCE				

Services de compression adaptative et de gestion sécurisée basés sur le cloud pour les données de santé 3D[44]	*		*	*	Images médicales 3d,imagerie militaire et criminalistique judiciaire	un moteur de compression dynamique et adaptative sans perte d’images médicales 3D et d’un système Saas Cloud, basé sur le moteur susmentionné	un accès efficace, sécurisé et flexible aux ressources de santé qui doivent être gérées par des applications médicales permet aux appareils ayant des caractéristiques matérielles et logicielles totalement différentes et hétérogènes d’interagir entre eux,	nécessite des protocoles réseau complexes, ainsi que des techniques avancées de compression une puissance de traitement relativement limités

Compression d'images par Région D’Intérêt [45] .		*	*		Images médicales	compression d’image par régions d’intérêt en utilisant la transformée cosinus discrète (TCD) et la quantification vectorielle (QV).	Pour Une image médicale l’information clinique peut être concentrée dans une région de l’image alors que le reste de l’image peut être représenté avec une moindre qualité, ce qui pourrait fournir un codage d’images efficace et précis.	L’image n’est plus divisible par des zones. On doit chaque fois changer la segmentation des zones selon nous besoins.

Medical Image Compression Using MapReduce [46]	*		*	*		L'idée principale est de diviser l'image en plusieurs parties (mappage), puis d'appliquer un algorithme de compression spécifique à chaque partie individuelle. Ensuite, dans la phase de réduction, les parties compressées sont combinées pour former l'image compressée finale.	traiter différentes parties de l'image en parallèle, ce qui peut accélérer le processus de compression. traiter de grandes quantités de données	La mise en œuvre d'un algorithme de compressin peut être complexe un surcoût en termes de communication réseau.

Image Compression Technique Using MapReduce for Satellite Image Processing [47]		*	*	*	tous les formats d’image.	une méthode de compression d'images satellite en utilisant le framework MapReduce. Le processus commence par la division de l'image en blocs, puis chaque bloc est traité individuellement. L'algorithme de compression spécifique est appliqué à chaque bloc, puis dans la phase de réduction, les blocs compressés sont combinés pour former l'image compressée finale.	une répartition équilibrée de la charge de travail. Inconvénients : le processus de compression et le rendre scalable pour les grandes images satellite.	Complexité : La mise en œuvre d'un algorithme Complexité : La mise en œuvre d'un algorithme

7. Conclusion :

En conclusion de ce chapitre nous avons exploré les différentes techniques de compression d'images, les différents types de compression et leurs applications. Le chapitre suivant se concentrera sur l'utilisation de l'architecture Hadoop/MapReduce pour la compression d'images, en mettant en œuvre des algorithmes tels que RLE (Run-Length Encoding) et Huffman.

CHAPITRE 2 : CONCEPTION

1. Introduction

Ce chapitre de conception se concentre sur l'implémentation détaillée des algorithmes de compression d'image RLE et Huffman en utilisant Hadoop. Le Run-Length Encoding (RLE) permet de représenter les séquences répétitives de pixels, tandis que l'algorithme de Huffman utilise un arbre binaire pour compresser les symboles en fonction de leur fréquence d'apparition. Nous explorerons les phases de conception, les défis spécifiques et les solutions proposées pour une compression d'image efficace en environnement distribué. Cette approche tirera parti de la puissance de la programmation parallèle et de la distribution des tâches offertes par Hadoop MapReduce pour améliorer la compression d'image.

2. RLE en Hadoop

RLE est une méthode simple mais efficace qui consiste à représenter les séquences répétitives de pixels dans une image par une paire . Cette approche est particulièrement adaptée pour compresser des images comportant de grandes zones de pixels identiques(le cas d'une image binaire" bitmap" avec laquelle nous avons travaillé).

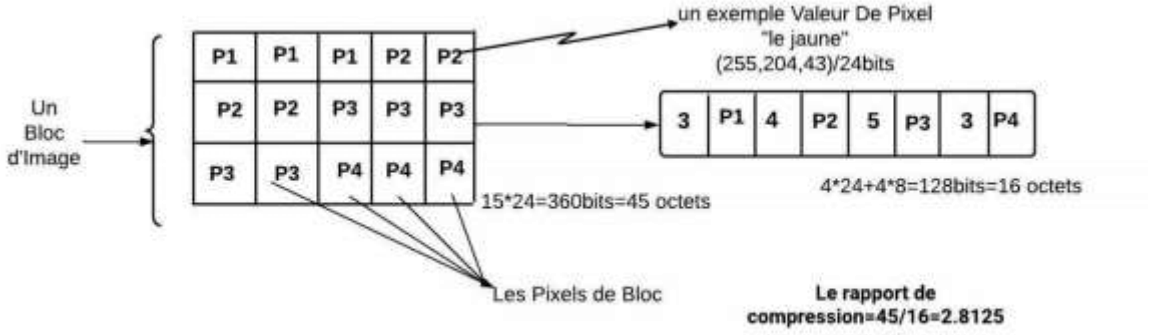

Figure 10 : Algorithme RLE sur un bloc d'image

Il est également important de noter que, dans le contexte des deux algorithmes de compression, RLE et Huffman, l'entrée de la phase Map est constituée d'un bloc d'image

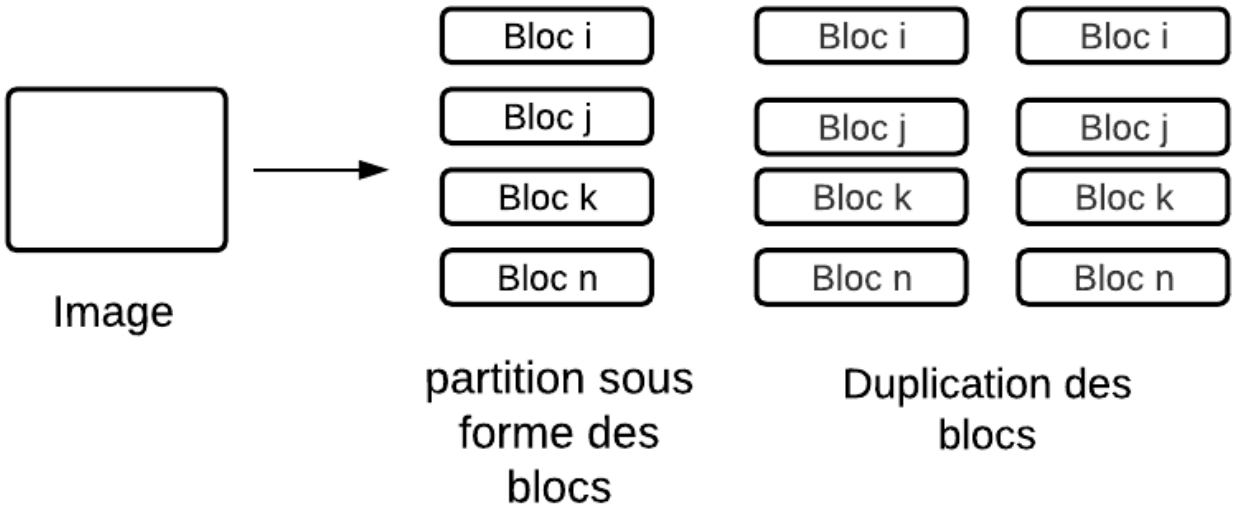

Figure 11 :Illustration de la Division des Blocs d'Image en Lignes dans HDFS

2.1. Phase Map

Dans cette phase : L'entrée est un bloc d'image sous forme de clé-valeur. La clé d'entrée de type LongWritable représente la position ou l'index de l'image dans le flux d'entrée, tandis que la valeur d'entrée de type BytesWritable contient les octets de l'image à traiter.

Une boucle est effectuée sur les pixels de l'image en comparant chaque pixel avec le pixel précédent pour détecter les séquences consécutives de pixels identiques. Une variable " StringBuilder" est utilisée pour construire la séquence compressée de blocs. Si un pixel est identique au pixel précédent, le compteur est incrémenté. Sinon, le pixel précédent et son nombre d'occurrences sont ajoutés à la séquence compressée.

Une fois la boucle terminée, le dernier pixel et son nombre d'occurrences sont ajoutés à la séquence compressée. Enfin, la méthode write est appelée pour émettre la clé d'entrée d'origine et la séquence compressée de blocs en tant que paire clé-valeur de sortie. La clé de sortie est identique à la clé d'entrée, et la valeur de sortie est de type Text et contient la séquence compressée de blocs.

Exemple :

Voici un exemple de déroulement du code RLE Mapper pour une image d'entrée :

- Supposons que nous ayons une image binaire de 8 pixels avec les valeurs suivantes : [5, 5, 3, 3, 3, 2, 2, 2]

- Le Mapper reçoit la clé d'entrée, qui représente l'index de l'image dans le flux d'entrée, et la valeur d'entrée, qui contient les octets de l'image à traiter.Les octets de l'image sont extraits et représentent les pixels de l'image : [5, 5, 3, 3, 3, 2, 2, 2](index de l'image, "[5, 5, 3, 3, 3, 2, 2, 2])

- Le Mapper itère sur les pixels de l'image et compare chaque pixel avec le pixel précédent.Lorsque le Mapper rencontre un pixel identique au pixel précédent, il incrémente le compteur de fréquence. Par exemple, lorsque le Mapper rencontre le deuxième pixel "5", le compteur est incrémenté et devient 2.

- Lorsque le Mapper rencontre un pixel différent du pixel précédent, il ajoute le pixel précédent et son nombre d'occurrences à la séquence compressée de blocs. Par exemple, lorsque le Mapper rencontre le premier pixel "3" après la séquence de pixels "5", il ajouté "5,2;" à la séquence compressée.Le Mapper continue à itérer sur les pixels et construit la séquence compressée de blocs comme suit : "5,2;3,3;2,3;"

- Une fois la boucle terminée, le dernier pixel "2" et son nombre d'occurrences "3" sont ajoutés à la séquence compressée.Le Mapper émet la clé d'entrée d'origine (l'index de l'image) et la séquence compressée de blocs en tant que paire clé-valeur de sortie. Par exemple, la paire de sortie serait : (index de l'image, "5,2;3,3;2,3;")

Figure 12:phase Map RLE de l'algorithme RLE

2.2. Phase Reduce

La classe RleReducer est une classe Reduce qui traite les données de sortie du Mapper pour effectuer une étape supplémentaire de réduction Les données de sortie du Mapper sont reçues sous forme de paires clé/valeur, où la clé est de type "LongWritable" et la valeur est de type" Text" Pour chaque valeur, la séquence compressée de blocs est extraite et ajoutée à un "StringBuilder" pour regrouper toutes les séquences compressées de l'image. Ensuite, la séquence compressée de blocs complète est écrite en sortie. Dans ce cas, la clé de sortie est une chaîne "RLE" et la valeur de sortie est la séquence compressée de tous les blocs complets de l'image .

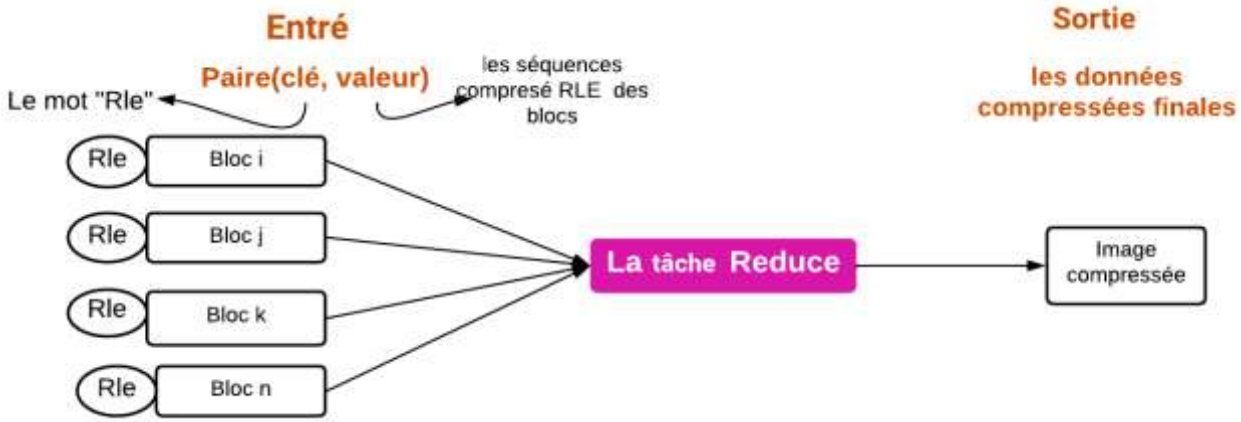

Figure 13:phase Reduce RLE de l'algorithme RLE

3. Huffman en Hadoop

Huffman en Hadoop/MapReduce est une technique de compression utilisée pour compresser efficacement les données d'image. L'algorithme de compression de Huffman est largement utilisé dans le domaine de la compression de données en raison de son efficacité et de sa simplicité. Lorsqu'il est appliqué à la compression d'image en utilisant le paradigme Hadoop/MapReduce, l'algorithme de compression de Huffman est exécuté de manière distribuée sur un cluster de nœuds.

Le fonctionnement de Huffman en Hadoop/MapReduce implique les étapes suivantes tel que on a travaillées avec deux jobs :

3.1 Huffman Job 1

3.1.1 Job 1 Phase Map

Lors de la phase Map du premier travail, l'entrée est constituée d'une série de paires clé-valeur. La clé représente un identifiant de bloc (ID de bloc), tandis que la valeur contient la séquence de pixels qui composent ce bloc. L'objectif initial de cette phase Map est de comptabiliser la fréquence d'occurrence de chaque pixel au sein de l'ensemble de l'image. Pour ce faire, chaque pixel est extrait de la séquence du bloc et émis sous forme d'une paire clé-valeur, où la clé représente le pixel individuel, et la valeur est initialement définie à 1, indiquant une occurrence unique.

Figure 14:La phase Map de premier travail dans l'algorithme Huffman en Hadoop

3.1.2 Job 1 Phase Reduce

La phase Réduce qui suit s'attèle à regrouper ces paires clé-valeur basées sur les pixels. Chaque tâche de réduction reçoit une clé, qui représente un pixel, et une liste de valeurs correspondante au nombre de fois que ce pixel a été rencontré dans différents blocs. Le rôle du Réducer est d'additionner ces valeurs pour obtenir la fréquence totale de chaque pixel dans l'image complète. Les résultats de cette phase de Réduction sont ensuite stockés sous forme de paires clé-valeur, où la clé représente le pixel et la valeur représente la fréquence de ce pixel par rapport à l'ensemble de l'image.

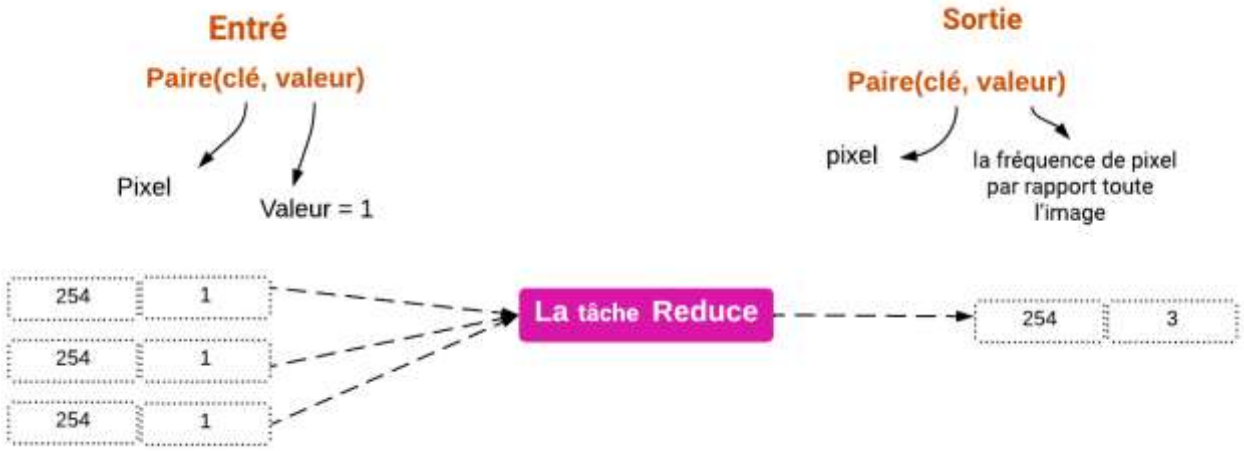

Figure 15: La phase Reduce du 1er travail de l'algorithme du Huffman en MapReduce

3.2 Huffman Job 2

Le deuxième Job, composé également de phases Map et Réduce, prend ces fréquences de pixels pour générer un dictionnaire de Huffman, également connu sous le nom d'arbre de Huffman.

3.2.1 Job 2 Phase Map

La phase Map de ce deuxième travail reçoit une entrée sous forme de paires clé-valeur, où la clé est l'identifiant de bloc, et la valeur est constituée des pixels du bloc avec leurs fréquences. Chaque tâche de map dans cette phase collecte les données de fréquence des pixels pour un bloc donné

Figure 16:La phase Map de la 2ème travail de l'algorithme du Huffman Job

3.2.2 Job 2 Phase Reduce

La phase de Réduction suivante est consacrée à la création du dictionnaire de Huffman en utilisant les fréquences cumulées des pixels. Une fois construit, ce dictionnaire est émis sous forme de paire clé-valeur.Pour chaque paire, la clé est symbolisée par le mot "Huff", indiquant clairement qu'il s'agit du dictionnaire de Huffman. La valeur est le dictionnaire lui-même.

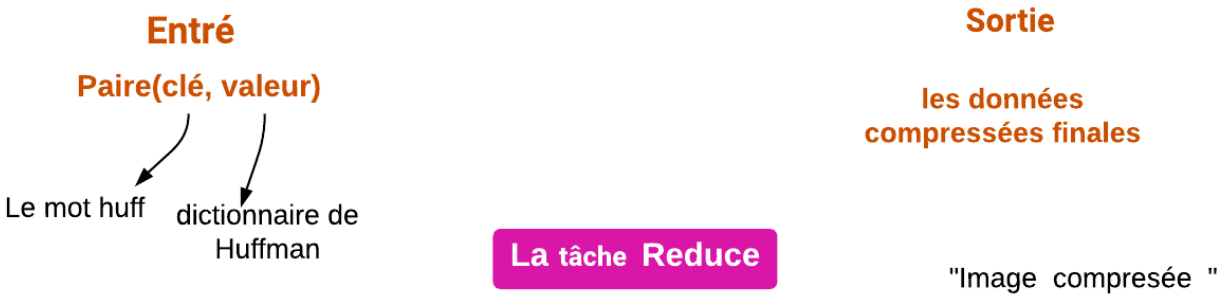

Figure 17:phase Reduce de la 2ème travail de l'algorithme du Huffman Job

En fin de compte, les données compressées finales, y compris le dictionnaire de Huffman, représentent la sortie finale du processus de compression. Cette conception en deux étapes garantit une compression efficace des images tout en tirant parti de la puissance de traitement parallèle offerte par Hadoop/MapReduce.

Le tableau ci-dessus compare le traitement séquentiel et le traitement parallèle (Hadoop) en termes de performance, de scalabilité, de temps de traitement, de gestion des grandes images, de complexité de mise en œuvre, de tolérance aux pannes, de coût et de flexibilité. Le traitement séquentiel est plus lent et moins évolutif, tandis que le traitement parallèle sur Hadoop offre de meilleures performances et une grande capacité de gestion des grandes images. Cependant, le traitement parallèle peut être plus complexe à mettre en œuvre en raison de la configuration du cluster Hadoop, mais il offre une plus grande souplesse pour

optimiser le processus de compression. De plus, le traitement parallèle sur Hadoop est plus résilient aux pannes grâce à la distribution des tâches sur plusieurs nœuds. En termes de coût

Critère de Comparaison	**Traitement Séquentiel**	**Traitement Parallèle (Hadoop)**
Performance	Plus lent pour de grandes images en raison du traitement séquentiel.	Meilleure performance avec le traitement parallèle distribué sur un cluster de nœuds Hadoop, surtout pour de grandes images.
Scalabilité	Moins scalable, car le traitement est limité à la capacité d'un seul nœud.	Scalablilité forte grâce à la capacité de traitement distribué des nœuds Hadoop. Peut gérer efficacement de grandes quantités de données d'image.
Temps de traitement	Plus long en raison du traitement séquentiel.	Réduit grâce au traitement parallèle simultané de plusieurs blocs d'image sur des nœuds Hadoop.
Gestion des grandes images	Peut rencontrer des limitations en raison de la taille de mémoire disponible sur un seul nœud.	Peut traiter efficacement de grandes images en répartissant le travail sur plusieurs nœuds Hadoop, en exploitant leur capacité de stockage et de traitement distribué.
Complexité de mise en œuvre	Relativement simple à implémenter en utilisant les algorithmes RLE et Huffman séquentiellement.	Implémentation plus complexe en raison de la mise en place d'un cluster Hadoop et de la gestion de la distribution des tâches et de la collecte des résultats

		intermédiaires.
Tolérance aux pannes	Vulnérable aux pannes d'un seul nœud, ce qui peut entraîner une perte de données ou des retards de traitement.	Résilient aux pannes grâce à la distribution des tâches sur plusieurs nœuds Hadoop. Si un nœud échoue, les autres peuvent continuer à travailler.
Coût	Moins coûteux en termes de matériel, mais peut nécessiter plus de temps de traitement.	Peut nécessiter un investissement initial plus important en raison de la mise en place d'un cluster Hadoop. Cependant, il peut être plus rentable en termes de performances et de temps de traitement pour de grandes quantités de données d'image.
Flexibilité	Le traitement séquentiel offre moins de flexibilité en termes de gestion des tâches et de configuration du processus de compression.	Le traitement parallèle dans Hadoop offre une grande flexibilité en permettant de configurer divers aspects du processus de compression, tels que le nombre de tâches, la répartition des données, etc. Cela permet d'optimiser les performances et de répondre aux exigences spécifiques du traitement.

Tableau 2.1: Comparaison entre le traitement séquentiel et parallèle de la compression d'image avec les algorithmes RLE et Huffman.

4. Conclusion

Ce chapitre pose les bases conceptuelles de notre projet, en fournissant une vision claire de la compression d'image en utilisant l'approches parallèles Hadoop /MapReduce.

Dans le chapitre suivant, nous parlons des outils utilisés pour réaliser notre projet avec les résultats obtenus et l'évaluation de notre travail.

CHAPITRE 3 : IMPLEMENTATION

1. Introduction

Dans ce chapitre, nous avons évalué les performances de deux techniques de compression, à savoir la compression RLE (Run-Length Encoding) et la compression Huffman, en utilisant deux modes différents : séquentiel et parallèle avec le framework Hadoop et le modèle MapReduce. Nous avons mesuré le taux de compression et le temps d'exécution pour chaque technique dans chaque mode. Les résultats obtenus nous permettent de comprendre les avantages et les limites de chaque approche et de tirer des conclusions sur les performances globales.

2. Environnement matériel et de développement utilisé

Le matériel utilisé pour développer notre méthode :
Un ordinateur portable avec les caractéristiques suivantes : Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz,Mémoire RAM de 8 GO Le système d'exploitation Windows 10 pro.
Type de système : système d'exploitation 64 bits.

2.2. Environnement de développement

2.2.1. Langage de programmation JAVA

Afin d'assurer que notre programme fournira les mêmes résultats s'il sera exécuté
En différents environnements d'essai. Nous avons choisi de réaliser notre application
Avec le langage Java.

2.2.2. Environnement de développement Netbeans

Nous avons réalisé notre travail en utilisant L'IDE NetBeans version 16 .C'est un environnement de développement et un outil pour les programmeurs pour écrire, compiler, déboguer et déployer des programmes.

3. implémentation

3.1. Hadoop framework

a) Démarrer les services Hadoop en exécutant quelques commandes.

Figure 18: Résultat de démarrage de Yarn et dfs

b) Vérification de l'état des services Hadoop :

Figure 19: Résultat de jps

c) Vérification de L'état de notre cluster Hadoop :

Figure 20 : L'état de notre cluster Hadoop

d) Compilation :

```
compile:
Created dir: C:\Users\user\Downloads\huffman\dist
Copying 1 file to C:\Users\user\Downloads\huffman\build
Copy libraries to C:\Users\user\Downloads\huffman\dist\lib.
Building jar: C:\Users\user\Downloads\huffman\dist\huffman.jar
To run this application from the command line without Ant, try:
java -jar "C:\Users\user\Downloads\huffman\dist\huffman.jar"
deploy:
jar:
BUILD SUCCESSFUL (total time: 7 seconds)
```

Figure 21: Résultat de création de « .jar »

e) mettre en place les images à compresser :

Figure 22: Les Blocs D'image

f) Lancement :

execution rle :

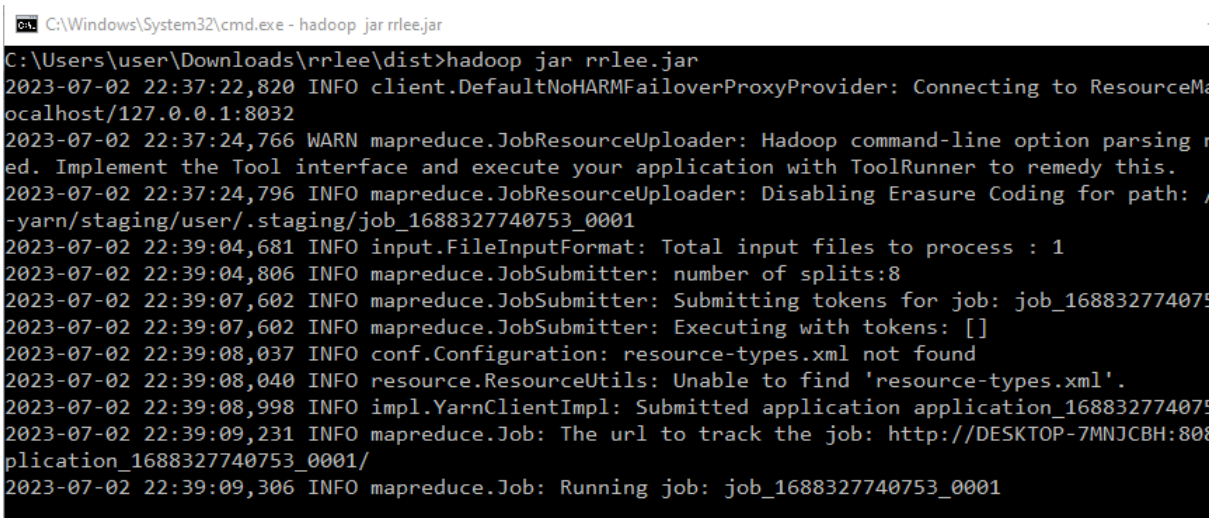

Figure 23 Résultat de l'exécution du job

```
2023-07-02 22:41:46,429 INFO mapreduce.Job:  map 0% reduce 0%
2023-07-02 22:42:53,777 INFO mapreduce.Job:  map 13% reduce 0%
2023-07-02 22:42:56,098 INFO mapreduce.Job:  map 25% reduce 0%
2023-07-02 22:42:57,384 INFO mapreduce.Job:  map 38% reduce 0%
2023-07-02 22:42:58,551 INFO mapreduce.Job:  map 75% reduce 0%
2023-07-02 22:42:59,609 INFO mapreduce.Job:  map 88% reduce 0%
2023-07-02 22:43:20,706 INFO mapreduce.Job:  map 100% reduce 0%
2023-07-02 22:43:23,773 INFO mapreduce.Job:  map 100% reduce 100%
2023-07-02 22:43:32,992 INFO mapreduce.Job: Job job_1688327740753_0001 completed successfully
2023-07-02 22:43:33,251 INFO mapreduce.Job: Counters: 55
        File System Counters
                FILE: Number of bytes read=1102
                FILE: Number of bytes written=2396254
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=17142
                HDFS: Number of bytes written=1056
                HDFS: Number of read operations=29
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Killed map tasks=1
                Launched map tasks=8
                Launched reduce tasks=1
                Data-local map tasks=8
                Total time spent by all maps in occupied slots (ms)=497168
                Total time spent by all reduces in occupied slots (ms)=25156
                Total time spent by all map tasks (ms)=497168
                Total time spent by all reduce tasks (ms)=25156
                Total vcore-milliseconds taken by all map tasks=497168
                Total vcore-milliseconds taken by all reduce tasks=25156
```

Figure 24: Résultat de l'exécution du job.

execution huffman :

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user\Downloads\huffman\dist>hadoop jar huffman.jar
2023-07-03 00:12:17,503 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceMana
.0.1:8032
2023-07-03 00:12:19,755 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not
the Tool interface and execute your application with ToolRunner to remedy this.
2023-07-03 00:12:19,798 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tm
user/.staging/job_1688327740753_0002
2023-07-03 00:14:02,918 INFO input.FileInputFormat: Total input files to process : 1
2023-07-03 00:14:03,029 INFO mapreduce.JobSubmitter: number of splits:8
2023-07-03 00:14:05,585 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1688327740753_
2023-07-03 00:14:05,588 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-07-03 00:14:06,091 INFO conf.Configuration: resource-types.xml not found
2023-07-03 00:14:06,104 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-07-03 00:14:06,381 INFO impl.YarnClientImpl: Submitted application application_1688327740753_
2023-07-03 00:14:06,558 INFO mapreduce.Job: The url to track the job: http://DESKTOP-7MNJCBH:8088/
327740753_0002/
2023-07-03 00:14:06,606 INFO mapreduce.Job: Running job: job_1688327740753_0002
```

```
2023-07-03 00:16:25,781 INFO mapreduce.Job:  map 0% reduce 0%
2023-07-03 00:17:24,882 INFO mapreduce.Job:  map 25% reduce 0%
2023-07-03 00:17:33,796 INFO mapreduce.Job:  map 38% reduce 0%
2023-07-03 00:17:35,936 INFO mapreduce.Job:  map 63% reduce 0%
2023-07-03 00:17:37,036 INFO mapreduce.Job:  map 88% reduce 0%
2023-07-03 00:18:01,149 INFO mapreduce.Job:  map 100% reduce 0%
2023-07-03 00:18:04,210 INFO mapreduce.Job:  map 100% reduce 100%
2023-07-03 00:18:11,338 INFO mapreduce.Job: Job job_1688327740753_0002 completed
2023-07-03 00:18:11,611 INFO mapreduce.Job: Counters: 55
        File System Counters
                FILE: Number of bytes read=25751
                FILE: Number of bytes written=2445435
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=17142
                HDFS: Number of bytes written=25649
                HDFS: Number of read operations=29
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Killed map tasks=2
                Launched map tasks=9
                Launched reduce tasks=1
                Data-local map tasks=9
                Total time spent by all maps in occupied slots (ms)=479360
                Total time spent by all reduces in occupied slots (ms)=33922
                Total time spent by all map tasks (ms)=479360
                Total time spent by all reduce tasks (ms)=33922
                Total vcore-milliseconds taken by all map tasks=479360
                Total vcore-milliseconds taken by all reduce tasks=33922
                Total megabyte-milliseconds taken by all map tasks=588654080
```

Figure 25: résultat d'exécution d'Huffman

4. Résultats des tests

	Séquentielle		Parallèle	
	Taux de compression	Temps d'exécutions (ms)	Taux de compression	Temps d'exécutions (ms)
RLE	0.037%	5.5	6%	326213
HUFFMAN	2.7%	29.2	157%	401300

Tableau 3.1 : Un tableau comparatif entre une image noir compressé par deux approches.

	Séquentielle		Parallèle	
	Taux de compression	Temps d'exécutions (ms)	Taux de compression	Temps d'exécutions (ms)
RLE	0.037%	5.5	7%	326213
HUFFMAN	2.7%	29.2	159%	401300

Tableau 2.2: Un tableau comparatif entre une image noir et blanc compressé par deux approches.

5. Discussion des résultats de la méthode MapReduce Parallèle :

- **Introduction**

Dans cette partie, nous avons évalué les performances de deux techniques de compression, à savoir la compression RLE (Run-Length Encoding) et la compression Huffman, en utilisant deux modes différents : séquentiel et parallèle avec le framework Hadoop et le modèle MapReduce. Nous avons mesuré le taux de compression et le temps d'exécution pour chaque technique dans chaque mode. Les résultats obtenus nous permettent de comprendre les avantages et les limites de chaque approche et de tirer des conclusions sur les performances globales.

- **Compression séquentielle**

La compression séquentielle consiste à appliquer les techniques de compression sur l'image en mode séquentiel, c'est-à-dire en traitant un seul bloc d'image à la fois.

- Pour l'image noir complet, nous avons obtenu les résultats suivants :

Technique : Huffman, RLE
Taux de compression : 2.7%, 0.037%
Temps d'exécution : 0.0055 secondes, 0.0292 secondes

5.1. Discussion des résultats pour l'image noir complet :

La compression séquentielle avec la technique Huffman a donné un taux de compression de 2.7%, ce qui signifie que la taille de l'image compressée représente seulement 2.7% de la taille de l'image originale. Cependant, le taux de compression obtenu avec la technique RLE est plus élevé, à 0.037%, ce qui indique une réduction plus importante de la taille de l'image compressée. En termes de temps d'exécution, la compression séquentielle avec Huffman a été plus rapide, prenant seulement 0.0055 secondes, tandis que la compression séquentielle avec RLE a pris 0.0292 secondes.

- Pour l'image demi noire et demi blanc, les résultats sont les suivants :

Technique : Huffman, RLE
Taux de compression : 103%, 0.28%
Temps d'exécution : 0.0095 secondes, 0.029 secondes

5.2. Discussion des résultats pour l'image demi noir et demi blanc :

La compession séquentielle avec la technique Huffman a donné un taux de compression de 103%, ce qui signifie que la taille de l'image compressée est supérieure à la taille de l'image originale, ce qui indique qu'il n'y a pas eu de compression réelle. Cela peut se produire si l'image est déjà fortement compressée ou si la technique de compression n'est pas adaptée aux caractéristiques spécifiques de l'image. Dans ce cas, la méthode Huffman n'a pas pu réduire la taille de l'image de manière significative, voire l'a augmentée légèrement. Cependant, le taux de compression obtenu avec la technique RLE est plus élevé, à 0.037%, ce qui indique une réduction plus importante de la taille de l'image compressée. En termes de temps d'exécution, la compression séquentielle avec huffman a été plus rapide, prenant seulement 0.0055 secondes, tandis que la compression séquentielle avec RLE a pris 0.0292 secondes.

- **Compression parallèle**

La compression parallèle utilise le framework Hadoop et le modèle MapReduce pour effectuer la compression en parallèle, ce qui permet de diviser le traitement sur plusieurs nœuds et d'accélérer les performances.

- Pour l'image noir complet, nous avons obtenu les résultats suivants :

Technique : Huffman, RLE
Taux de compression : 157%, 6%
Temps d'exécution : 401.3 secondes, 326.213 secondes

Discussion des résultats pour l'image noir complet :

Le taux de compression élevé de 157% ce qui signifie que la taille de l'image compressée est supérieure à la taille de l'image originale, ce qui indique qu'il n'y a pas eu de compression réelle à cause des mêmes problèmes déjà mentionné. Cependant, le taux de compression obtenu avec la technique RLE est plus élevé, à 0.28%, ce qui indique une réduction plus importante de la taille de l'image compressée. En termes de temps d'exécution, la compression séquentielle avec Rle a été plus rapide, prenant seulement 326 secondes, tandis que la compression séquentielle avec Hoffman a pris 401 secondes.

- Pour l'image noir complet, nous avons obtenu les résultats suivants :

Technique : Huffman, RLE
Taux de compression : 159%, 7%
Temps d'exécution : 401.3 secondes, 326.213 secondes

Discussion des résultats pour l'image Demi noir et demi blanc :

Le taux de compression 159% ce qui signifie que la taille de l'image compressée est supérieure à la taille de l'image originale, ce qui indique qu'il n'y a pas eu de compression réelle à cause des mêmes problèmes déjà mentionné. Cependant, le taux de compression obtenu avec la technique RLE est plus élevé, à 7% de la taille de l'image originale, ce qui indique une réduction plus importante de la taille de l'image compressée. En termes de temps d'exécution, la compression séquentielle avec Rle a été plus rapide, prenant seulement 326 secondes, tandis que la compression séquentielle avec Hoffman a pris 401 secondes.

5.3. Comparaison des résultats

En comparant les résultats de la compression séquentielle et parallèle, nous pouvons observer les différences suivantes :

- **Taux de compression** : Les deux approches (séquentielle et parallèle) montrent des taux de compression élevés pour la technique Rle. Cela est dû à la capacité de la technique Rle à représenter les symboles fréquents dans l'image avec des codes courts. Cela permet de réduire considérablement la taille de l'image compressée si elle contient de redondance.

- **Temps d'exécution :** La compression séquentielle est beaucoup plus rapide que la compression parallèle. Cela s'explique par le fait que la compression séquentielle traite un seul bloc d'image à la fois, tandis que la compression parallèle nécessite des opérations supplémentaires de division et de combinaison des blocs d'image, ce qui ajoute une surcharge et prolonge le temps d'exécution. De plus, l'utilisation d'un seul nœud (notre ordinateur) pour l'exécution du **MapReduce** job limite les performances et la vitesse d'exécution, et aussi l'opération **MapReduce** est très couteuse en terme de ressources et de temps d'exécution, et aussi la capacité de notre machine déjà mentionné en aux joue un rôle essentiel pour les résultats obtenus.

6. **Conclusion**

En conclusion, la compression séquentielle avec la technique RLE offre des taux de compression élevés pour les deux types d'images. Cependant, la compression parallèle introduit des coûts supplémentaires en termes de temps d'exécution en raison de la surcharge liée à l'utilisation de Hadoop et au traitement **MapReduce**. Les résultats et les graphiques générés nous permettent de mieux comprendre les performances des différentes approches et de prendre des décisions informées en fonction des exigences spécifiques de compression.

CONCLUSION GENERALE

CONCLUSION GENERALE

Dans notre travail, nous nous sommes focalisés sur le problème de la compression d'image en utilisant des modèles de programmation parallèles Hadoop (MapReduce). Nous avons examiné le potentiel du framework Hadoop avec ses composants MapReduce et HDFS pour développer une puissante solution de compression d'image.
Notre démarche repose sur l'utilisation de l'algorithme de compression basé sur la méthode MapReduce. Nous avons adapté des modèles de programmation parallèles pour gérer les différents stades du processus de compression d'image, En outre, en utilisant le système de fichiers HDFS réparti de Hadoop. Cela permet de gérer les données de façon efficace à grande échelle.

Les résultats obtenus pour les deux approches, séquentielle et parallèle, ont révélé des rapports de compression élevés au moyen de la technique RLE. Cependant, il convient de noter que la compression séquentielle a montré une vitesse d'exécution beaucoup plus rapide que la compression parallèle avec un seul nœud (notre ordinateur) pour l'exécution du travail MapReduce.

Comme perspective de ce travail, nous pensons que ça cera intéressant de combiner l'algorithme RLE et Huffman, cette combinaison offre des perspectives intéressantes pour une meilleure compression d'image. Cette approche permettrait de tirer profit des avantages de chaque algorithme et d'obtenir une compression plus efficace en exploitant les propriétés de redondance et de répartition des symboles sur l'image.

REFERENCE

Références

[1]J. Liu, S. Wang, and R. Urtasun, ”Dsic: Deep stereo image compression,” in 2019 IEEE/CVF Interna- tional Confer

ence on Computer Vision (ICCV), 2019, pp. 3136–3145.

[2]J. L. Nunez and S. Jones, ”Run-length coding extensions for high performance hardware data compres- sion,” in IEE

Proceedings-Computers and Digital Techniques, vol. 150, no. 6, pp. 387–395, 2003, doi: 10.1049/ip-cdt:20030750.

[3]D. T. V˜o. S. Lertrattanapanich and Y. Kim, ”Visually lossless compression for color images with low line memory

requirement using non-uniform quantizers,” 2011 IEEE International Conference on Consumer Electronics (ICCE),

2011, pp. 639-640, doi: 10.1109/ICCE.2011.5722783.

[4]S. D. Rane and G. Sapiro, ”Evaluation of JPEG-LS, the new lossless and controlled-lossy still image compression

standard, for compression of high-resolution elevation data,” in IEEE Transactions on Geoscience and Remote Sens

ing, vol. 39, no. 10, pp. 2298-2306, Oct. 2001, doi: 10.1109/36.957293.

[5]J. Liu, G. Lu, Z. Hu, and D. Xu, ”A unified end-to-end framework for efficient deep image compression,” arXiv

preprint arXiv:2002.03370, 2020.

[6] C. Raghavendra, S. Sivasubramanian, and A. Kumaravel, ”Improved image compression using effective lossless

compression technique,” Cluster Computing, vol. 22, no. 2, pp. 3911–3916, 2019, doi: 10.1007/s10586-018-2508-1.

[7] Y. Zhang and D. A. Adjeroh, ”Prediction by Partial Approximate Matching for Lossless Image Compression,” in

IEEE Transactions on Image Processing, vol. 17, no. 6, pp. 924-935, June 2008, doi: 10.1109/TIP.2008.920772.

[8] I. Jumakulyyev and T. Schultz, ”Lossless PDE-based Compression of 3D Medical Images,” in SSVM, 2021. pp.

450-462.

[9] S.benfriha et S.hamel, « Segmentation d'image par Coopération région-contours », Mémoire Master Université KasdiMerbah-Ouargla Professionnel, Domaine : Informatique et Technologiede l'Information, 2016.

[10] Rafael C. Gonzalez et Richard E.Woods, « Digital image processing », article Bibliothèque du Congrès Catalogage-en-publication2008

[11] Didier Muller, « Traitement d'images », cours, 2juin 2016.

[12] S.benfriha et S.hamel, « Segmentation d'image par Coopération région-contours », Mémoire Master Université KasdiMerbah-Ouargla Professionnel, Domaine : Informatique et Technologiede l'Information, 2016.

[13] Edmond Femandez, « Caracteristiques techniquesdes images numeriques », Archives Nationales. Outre-Mer, avril 2010.

[14] Ibrahim Guelzim, « Contributions aux traitements d'images perspectives et omnidirectionnelles par des outils statistiques », thèse de doctorat faculté des sciences rabat — maroc, informatiques et télécommunications, 12 mai 2012.

[15]Barni, M. (Ed.). (2006). Document and Image Compression (1st ed.). CRC Press. https://doi.org/10.1201/9781315221298

[16]Rabbani, H., & Joshi, R. C. (2005). An overview of the JPEG2000 still image compression standard. Signal Processing: Image Communication, 19(5), 395-412. DOI: 10.1016/j.image.2004.12.004.

[17] N. G. Kingsbury, The dual-tree complex wavelet transform : a new efficient tool for imagerestoration and enhancement,_ in European Signal Processing Conference,Sept. 1998, pp. 319_322.

[18] Abu Taleb, S. A., Musafa, H. M., Khtoom, M., &Gharaybih, I. K. (2010). ImprovingLZW Image Compression. In European Journal of Scientific Research (Vol. 44, Issue3). http://www.eurojournals.com/ejsr.htm.

19]https://www.techno-science.net/dossier/compression-images-numeriques-D9-page-9.html#:~:text=La%20compression%20d'une%20image,sauvegarder%20qu'une%20seule%20fois.

[20]. N. Moreau. Techniques de compression des signaux. Masson, Paris, 1995.,

[21]2David S. Taubman and Michael W. Marcellin. JPEG 2000 : Image Compression letmdamentals, Standards and Practice. Kluwer Academic Publishers, Norwell, MA, USA, 2001.].

[22]http://fr.wikipedia.org/wiki/JPEG_2000

[23]https://kinsta.com/fr/blog/compression-avec-ou-sans-perte/

[24] DAVID A. HUFFMAN. A Method of the Construction of Minimum Redundancy Codes. The Institute of Radio Engineers, volume 40, No. 9, pp. 1098_1101, 1952.

[25] M. SAHIR, « Compression des images numériques par la technique des ondelettes »,Mémoire de Magister, soutenu le 19 juin 2011 à l'Université de Sétif.

[26] E. LE PENNEC, «Compression d'image» — Images des Mathématiques, CNRS, 2006.

[27]J.-N. GOUYET, C. NELSON et M. LEGER, « JPEG 2000, format de compression d'image numérique par ondelettes – Principes », Techniques de l'Ingénieur, 2016.

[28]Paul G.Howard and Jeffery Scott Vitter .Arithmetic Coding for Data Compression. Technical Report Technical report, DUKE-TR-1994-09, 1994

[29]ETHW. "milestones :lempel-ziv data compression algorithm, 1977". https: //ethw. org/Milestones : Lempel- Ziv Data_Compression_ Algorithme _1977, september 2004.

[30]Abu Taleb, S. A., Musafa, H. M., Khtoom, M., &Gharaybih, I. K. (2010). ImprovingLZW Image Compression. In European Journal of Scientific Research (Vol. 44, Issue 3). http://www.eurojournals.com/ejsr.htm.

[31]https://www2.ulb.ac.be/cours/acohen/travaux_2006_infodoc/CompressionNumerique/TypeDonneesImageJPEG2000.htm.

[32] R. Souadnia & K. Benmahamed, "Compression des Images Numériques Fixes Par les Fractales". Proceeding FRACTALES' 2000, Université Mentouri Constantine, Algérie, Novembre 2000.

[33] A. Ali-Pacha & N. Hadj-Said, "Compression Des Images Fixes Par Fractale: Partitionnement Quadtree". Proceeding FRACTALES 2000, Université Mentouri Constantine, Algérie, Novembre 2000.

[34] Hadoop. http://hadoop.apache.org/

[35]//Cloud_Hadoop_MapReduce_For_Remote_Sensing_Image_Analysis-with-cover-page-v2 (1)

[36]Hadoop and solid state drives, https://www.facebook.com/hadoopfs, Accessed: 2019-09-29.

[37] The present and future of apache hadoop: A community meetup at linkedin, https://engineering.linkedin.com/blog/2019/02/the- presentand-future-of-apache-hadoop--a-community-meetup-at-l, Accessed: 2019-09-29.

[38] Hadoop at twitter, https://blog.twitter.com/engineering/en_us/a/2010/hadoop-at-twitter.html, Accessed: 2019-09-29.

[39]How twitter, facebook & ny times use hadoop? https : / / medium . com /@suchitmajumdar/how-twitter-facebook-ny-times-use-hadoop-782b5d8940cf, Accessed: 2019-09-29.

[40] What is HDFS? Hadoop Distributed File System overview

[41]"MapReduce: Simplified Data Processing on Large Clusters" par Jeffrey Dean et Sanjay Ghemawat, publié en 2004 dans le livre Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI '04).

[42] Charles Bouillaguet, Systèmes et Traitements Répartis

[43] MapReduce : traitement simplifié des données sur les grands clusters

[44] A. Castiglione, R. Pizzolante, A. De Santis, B. Carpentieri, A. Castiglione, F. Palmieri, "Cloud-based adaptive compression and secure management services for 3D healthcare data," Future Generation Computer Systems, Volumes 43-44, 2015, Pages 120-134, ISSN 0167-739X, DOI: 10.1016/j.future.2014.07.001L

[45] Jordane Dorne. Représentation sémantique de données géospaciales au service de l'analyse de changements. Traitement des images [eess.IV]. Université Toulouse le Mirail - Toulouse II, 2021. Français. ⟨NNT : 2021TOU20068⟩. ⟨tel-03618363⟩

[46] Chen, M., Lu, K., & Zhang, L. (2013). Medical Image Compression Using MapReduce. In 2013 IEEE International Conference on Bioinformatics and Biomedicine (pp. 223-226). IEEE. DOI: 10.1109/BIBM.2013.6732497

[47] Chauhan, M., Agrawal, S., & Singh, R. (2017). Image Compression Technique Using MapReduce for Satellite Image Processing. In 2017 7th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 318-323). IEEE. DOI: 10.1109/CONFLUENCE.2017.7942712.