UE Software Engineering

Software Engineering Project 2025

compressing for speed up transmission

Préparé par:

Abdelbaki Kacem

2025

*C*hapitre *1*

Compression de Tableaux d'Entiers par Empaquetage de Bits

Plan

1.1 Introduction

Ce rapport introduit le contexte général du projet, y compris une analyse de l'état actuel, l'énoncé du problème, la solution proposée et la méthodologie choisie pour planifier et exécuter le projet.

1.2 Contexte du Projet

Le projet consiste en la conception et l'implémentation d'un système de compression pour des tableaux d'entiers. L'objectif principal est de réduire la taille des tableaux d'entiers pour une transmission plus efficace sur un réseau, tout en préservant la capacité d'accéder rapidement à n'importe quel élément du tableau compressé.

1.3 Problème Cible

La transmission brute de tableaux d'entiers (typiquement 32 bits par entier) peut être très inefficace, surtout lorsque les valeurs réelles contenues dans le tableau peuvent être représentées avec beaucoup moins de bits. Le défi est de trouver une méthode de compression qui :

- Réduise la quantité totale de bits à transmettre.
- Permette un accès direct rapide (fonction get(i)) à n'importe quel élément sans avoir à décompresser l'ensemble du tableau.
- Gère efficacement les tableaux contenant des entiers de largeurs de bits variables, en particulier lorsque quelques grandes valeurs pourraient imposer une largeur de compression inefficace pour tous les autres.

1.4 Solution Proposée

La solution implémentée comprend trois stratégies de compression distinctes basées sur l'empaquetage de bits :

- **Sans Fractionnement (No Spill) :** Empaquette les entiers dans des blocs de 32 bits sans permettre à un seul entier de traverser la limite d'un bloc.
- **Avec Fractionnement (Spill) :** Empaquette les entiers séquentiellement dans un flux binaire, permettant à un seul entier de traverser les limites des blocs de 32 bits pour une meilleure densité.

- **Avec Débordement (Overflow) :** Identifie un seuil de bits. Les entiers dépassant ce seuil sont stockés dans une liste séparée. Le tableau principal contient des indicateurs compacts pointant vers cette liste de débordement.

Chaque stratégie est accompagnée de fonctions pour compresser, décompresser et accéder directement aux éléments. Une méthode de seuillage dynamique est implémentée pour optimiser la stratégie de débordement. Les performances sont mesurées et un seuil de bande passante est calculé pour déterminer à partir de quel moment la compression devient avantageuse.

compression devient avantageuse.

1.5 Méthodologie

Le projet a été conçu et implémenté en Java. La structure du code suit une approche modulaire, avec des fonctions distinctes pour chaque stratégie de compression et ses opérations associées (compresser, décompresser, get). Une énumération (CompressionType) est utilisée pour simuler un modèle d'usine (factory pattern) et sélectionner la méthode à exécuter. Les performances sont évaluées en mesurant les temps d'exécution de chaque fonction principale à l'aide de System.nanoTime() sur plusieurs itérations.

Conclusion

Dans ce projet, nous avons établi le contexte d'un projet de compression par compactage de bits. Nous avons exposé le problème que nous traitons et proposé une solution qui utilise trois stratégies distinctes. De plus, nous avons détaillé la méthodologie de mise en œuvre et la manière dont nous évaluerons les performances.