UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Ré-identification efficace dans les types de données répliquées sans conflit (CRDTs)

THÈSE

présentée et soutenue publiquement le TODO : Définir une date

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Matthieu Nicolas

Composition du jury

Président :	Stephan Merz	
Rapporteurs :	Le rapporteur 1	de Paris
	Le rapporteur 2	
	suite	taratata
	Le rapporteur 3	
Examinateurs :	L'examinateur 1	d'ici
	L'examinateur 2	
Membres de la famille :	Mon frère	
	Ma sœur	

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

Les remerciements.

Je dédie cette thèse
à ma machine.
Oui, à Pandore,
qui fut la première de toutes.

Sommaire

Table des figures

Introduction

Chapitre 1

État de l'art

Sommaire

1.1 Transformées opérationnelles

1.2 Séquences répliquées sans conflits

1.2.1 Type de données répliquées sans conflits

1.2.2 Approches pour les séquences répliquées sans conflits

1.2.3 Mitigation du surcoût des séquences répliquées sans conflits

Core-Nebula

LSEQ

1.3 LogootSplit

Chapitre 2

Présentation de l'approche

Sommaire

2.1 Modèle du système

2.2 Définition de l'opération de renommage

2.2.1 Objectifs

2.2.2 Propriétés

2.2.3 Contraintes

Chapitre 3

Renommage dans un système centralisé

Sommaire

3.1 RenamableLogootSplit

3.1.1 Opération de renommage proposée

3.1.2 Gestion des opérations concurrentes au renommage

3.1.3 Intégration d'une opération

3.1.4 Évolution du modèle de cohérence

3.1.5 Récupération de la mémoire des états précédents

3.2 Validation

3.2.1 Preuve de correction

3.2.2 Complexité temporelle

3.3 Discussion

3.3.1 Stockage d'anciens états sur disque

3.3.2 Compression de l'opération de renommage

3.3.3 Limitation de la taille de l'opération de renommage

3.4 Conclusion

Chapitre 4

Renommage dans un système distribué

Sommaire

4.1 RenamableLogootSplit v2

4.1.1 Conflits en cas de renommages concurrents

4.1.2 Méthode de résolution de conflits proposée

4.1.3 Relation de priorité entre renommages

4.1.4 Algorithme d'annulation de l'opération de renommage

4.1.5 Mise à jour de l'intégration d'une opération

4.1.6 Récupération de la mémoire des états précédents

4.2 Validation

4.2.1 Complexité temporelle

4.2.2 Expérimentations

Scénario d'expérimentation

Implémentation des simulations

4.2.3 Résultats

Convergence

Consommation mémoire

Temps d'intégration des opérations "simples"

Temps d'intégration de l'opération de renommage

4.3 Discussion

4.3.1 Implémentation alternative à base d'operation-log

4.3.2 Définition de relations de priorité plus optimales

4.3.3 Report de la transition vers la nouvelle epoch principale

4.4 Conclusion

Chapitre 5

Stratégies de déclenchement du renommage

Sommaire

5.1 Motivation

5.2 Stratégies proposées

5.2.1 Propriétés

5.2.2 Stratégie 1 : ? ? ?

5.2.3 Stratégie 2 : ? ? ?

NOTE : Peut considérer une stratégie où on prend en compte la hauteur de l'epoch tree pour déclencher un renommage : peut attendre qu'on ait plus qu'une epoch pour autoriser un nouveau renommage d'avoir lieu. Permettrait d'empêcher le cas où un noeud revient après 6 mois d'absence et doit intégrer 100 renommages avant de pouvoir collaborer. Mais dans le cas où un noeud ne rejoint plus la collaboration, bloque le mécanisme de renommage pour les autres

5.3 Évaluation

5.4 Conclusion

Chapitre 6

Conclusions et perspectives

Sommaire

6.1 Résumé des contributions

6.2 Perspectives

6.2.1 Définition de relations de priorité plus optimales

6.2.2 Redéfinition de la sémantique du renommage en déplacement d'éléments

6.2.3 Définition de types de données répliquées sans conflits plus complexes

Annexe A

Algorithmes

Résumé

Afin d'assurer leur haute disponibilité, les systèmes distribués à large échelle se doivent de répliquer leurs données tout en minimisant les coordinations nécessaires entre noeuds. Pour concevoir de tels systèmes, la littérature et l'industrie adoptent de plus en plus l'utilisation de types de données répliquées sans conflits (CRDTs). Les CRDTs sont des types de données qui offrent des comportements similaires aux types existants, tel l'Ensemble ou la Séquence. Ils se distinguent cependant des types traditionnels par leur spécification, qui supporte nativement les modifications concurrentes. À cette fin, les CRDTs incorporent un mécanisme de résolution de conflits au sein de leur spécification.

Afin de résoudre les conflits de manière déterministe, les CRDTs associent généralement des identifiants aux éléments stockés au sein de la structure de données. Les identifiants doivent respecter un ensemble de contraintes en fonction du CRDT, telles que l'unicité ou l'appartenance à un ordre dense. Ces contraintes empêchent de borner la taille des identifiants. La taille des identifiants utilisés croît alors continuellement avec le nombre de modifications effectuées, aggravant le surcoût lié à l'utilisation des CRDTs par rapport aux structures de données traditionnelles. Le but de cette thèse est de proposer des solutions pour pallier ce problème.

Nous présentons dans cette thèse deux contributions visant à répondre à ce problème : (i) Un nouveau CRDT pour Séquence, RenamableLogootSplit, qui intègre un mécanisme de renommage à sa spécification. Ce mécanisme de renommage permet aux noeuds du système de réattribuer des identifiants de taille minimale aux éléments de la séquence. Cependant, cette première version requiert une coordination entre les noeuds pour effectuer un renommage. L'évaluation expérimentale montre que le mécanisme de renommage permet de réinitialiser à chaque renommage le surcoût lié à l'utilisation du CRDT. (ii) Une seconde version de RenamableLogootSplit conçue pour une utilisation dans un système distribué. Cette nouvelle version permet aux noeuds de déclencher un renommage sans coordination préalable. L'évaluation expérimentale montre que cette nouvelle version présente un surcoût temporaire en cas de renommages concurrents, mais que ce surcoût est à terme.

Mots-clés: CRDTs, édition collaborative en temps réel, cohérence à terme, optimisation mémoire, performance

Abstract

Keywords: CRDTs, real-time collaborative editing, eventual consistency, memory-wise optimisation, performance