

Titre de la thèse (sur plusieurs lignes si nécessaire)

*Traduction du titre de la thèse (sur plusieurs lignes si
nécessaire)*

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°575, electrical, optical, bio : physics and engineering (EOBE)
Spécialité de doctorat : Électronique et optoélectronique, nano- et
microtechnologies

Graduate School : Sciences de l'ingénierie et des systèmes. Référent : Faculté des
sciences d'Orsay

Thèse préparée dans l'unité de recherche **C2N (Université Paris-Saclay, CNRS)**, sous la
direction de **Prénom NOM**, Professeur, le co-encadrement de **Prénom NOM**, Maître de
conférences

Thèse soutenue à Paris-Saclay, le JJ mois AAAA, par

Prénom NOM

Composition du jury

Membres du jury avec voix délibérative

Prénom NOM
Titre, Affiliation
Prénom NOM
Titre, Affiliation
Prénom NOM
Titre, Affiliation
Prénom NOM
Titre, Affiliation
Prénom NOM
Titre, Affiliation

Président ou Présidente
Rapporteur & Examineur / trice
Rapporteur & Examineur / trice
Examineur ou Examinatrice
Examineur ou Examinatrice

Titre : Titre de la thèse (sur plusieurs lignes si nécessaire)

Mots clés : premier mot clé, deuxième mot clé, troisième mot clé

Résumé : Le résumé présente en quelques pa- seule page.

graphes le contexte de la thèse, la question de recherche, la démarche suivie et les principaux résultats obtenus. Il est rédigé en français et tient, avec sa traduction en anglais, sur une

Le second paragraphe peut préciser la portée des résultats, leurs limites et les perspectives ouvertes par le travail.

Title : Traduction du titre de la thèse (sur plusieurs lignes si nécessaire)

Keywords : first keyword, second keyword, third keyword

Abstract : The abstract presents in a few para- with the French summary, on a single page.

graphs the context of the thesis, the research question, the approach taken and the main results. It is written in English and fits, together

The second paragraph may state the scope of the results, their limits and the perspectives the work opens.

Table des matières

Introduction	5
1 Édition collaborative de documents	7
1.1 Exigences relatives au système	7
2 Approches de la synchronisation	9
2.1 Transformation opérationnelle	9
2.1.1 Fonction de transformation	9
2.2 Types de données répliqués sans conflit	9
3 Conception et implémentation	11
3.1 Stockage de l'historique	11
4 Évaluation	13
4.1 Discussion des résultats	14
Conclusion	15
Bibliographie	17

Introduction

Aujourd'hui, les documents textuels sont rarement l'œuvre d'un seul auteur. Les étudiants rédigent leurs travaux de séminaire en binôme, les développeurs maintiennent ensemble la documentation technique et les équipes de recherche finalisent leurs articles à la dernière minute depuis plusieurs endroits à la fois. Pour qu'une telle écriture fonctionne, l'éditeur doit garantir que tous les participants voient le même texte et qu'aucune modification ne se perde.

L'objectif de ce travail est de comparer les deux principales approches de la synchronisation des documents lors de l'édition collaborative en temps réel, la transformation opérationnelle et les types de données répliqués sans conflit, et de les valider sur un prototype d'éditeur web. Le premier chapitre définit les exigences relatives à un tel système, le deuxième décrit les deux approches, le troisième la conception du prototype et le quatrième les résultats des mesures.

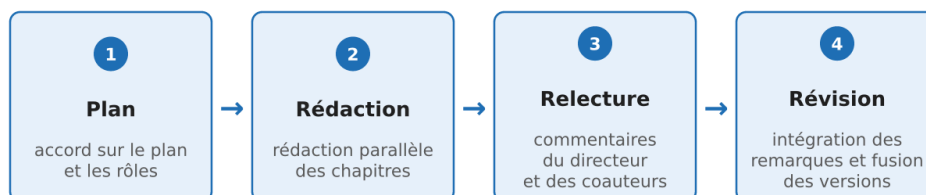
1 - Édition collaborative de documents

Par édition collaborative, nous entendons la situation dans laquelle plusieurs utilisateurs modifient le même document et où leurs modifications se propagent mutuellement sans fusion manuelle. Selon la rapidité avec laquelle les modifications se propagent, nous distinguons trois modes de travail fondamentaux :

- **l'édition asynchrone**, dans laquelle les auteurs s'échangent des versions complètes du document et ne les fusionnent qu'a posteriori ;
- **l'édition avec verrouillage**, où un seul auteur à la fois peut modifier une partie donnée du document ;
- **l'édition en temps réel**, dans laquelle chaque modification est immédiatement répercutée chez les autres utilisateurs.

Ce travail porte exclusivement sur le dernier mode. C'est le plus confortable pour les utilisateurs, mais il impose les plus fortes exigences à la conception du système, car les modifications naissent simultanément en de nombreux endroits et arrivent dans un ordre différent.

L'édition en temps réel ne change d'ailleurs pas seulement la technique, mais aussi le déroulement de toute la collaboration. Le processus typique d'écriture collaborative, tel que l'ont décrit les utilisateurs interrogés, est présenté à la Figure 1.1.



les étapes 2 à 4 se répètent tant que la relecture apporte des remarques substantielles

Figure 1.1 – Déroulement typique de l'écriture collaborative d'un document

1.1 . Exigences relatives au système

À partir d'entretiens avec des utilisateurs d'éditeurs existants et de la littérature spécialisée, nous avons dégagé quatre exigences auxquelles un système d'édition collaborative doit satisfaire :

1. la convergence, c'est-à-dire qu'une fois toutes les modifications remises, toutes les copies du document ont le même contenu ;

2. la préservation de l'intention, c'est-à-dire que le résultat d'une modification correspond à ce que l'auteur voyait au moment de l'effectuer ;
3. la réactivité, c'est-à-dire qu'une modification locale prend effet immédiatement, sans attendre le serveur ;
4. la robustesse, c'est-à-dire qu'une brève coupure de connexion n'entraîne pas la perte du texte en cours de rédaction.

Les deux premières exigences concernent la correction, les deux autres l'utilisabilité. La pratique montre que c'est précisément la combinaison d'une réactivité immédiate et d'une convergence garantie qui constitue la partie la plus difficile de toute la conception.

2 - Approches de la synchronisation

Les systèmes d'édition collaborative en temps réel appliquent d'abord chaque modification localement et ne la diffusent qu'ensuite aux autres participants. Ils diffèrent surtout par la manière dont ils traitent la situation où deux modifications naissent simultanément et où chaque participant les reçoit dans un ordre différent.

2.1 . Transformation opérationnelle

La transformation opérationnelle s'appuie sur les travaux d'Ellis et Gibbs [1], qui ont été les premiers à décrire comment adapter une opération concurrente de sorte qu'elle puisse être exécutée même après une autre opération dont son auteur n'avait pas connaissance. Chaque modification est enregistrée comme une opération d'insertion ou de suppression à une position donnée du texte.

Si, par exemple, un utilisateur insère un mot au début d'un paragraphe et qu'un autre supprime simultanément un mot à sa fin, la position de la suppression doit être décalée de la longueur du mot inséré. Ce décalage est calculé par la fonction de transformation.

2.1.1 . Fonction de transformation

La fonction de transformation reçoit une paire d'opérations concurrentes et renvoie leurs versions ajustées. Pour que toutes les copies du document convergent, la fonction doit satisfaire les propriétés dites de transformation, dont le respect est difficile à démontrer pour des opérations quelconques. La plupart des systèmes déployés s'appuient donc sur un serveur central qui détermine un ordre unique des opérations.

2.2 . Types de données répliqués sans conflit

La seconde approche est représentée par les types de données répliqués sans conflit, désignés collectivement par le sigle CRDT [2]. Au lieu d'ajuster les opérations a posteriori, elle choisit une représentation du document dans laquelle les opérations concurrentes commutent et leur ordre n'a aucune influence sur le résultat.

Un type de données répliqué est sans conflit si deux répliques quelconques ayant reçu le même ensemble de mises à jour ont le même état, quel que soit l'ordre dans lequel ces mises à jour leur ont été remises.

Chaque caractère du document reçoit un identifiant unique qui reste le même pendant toute la durée d'existence du caractère. La position d'insertion n'est alors pas déterminée par un nombre, mais par une référence à l'identifiant du caractère voisin, de sorte que des modifications concurrentes ailleurs dans le texte ne peuvent pas la décaler.

3 - Conception et implémentation

Le prototype d'éditeur se compose d'un client web, d'un serveur de synchronisation et d'une base de données dans laquelle le serveur enregistre en continu l'état des documents. Le client et le serveur sont écrits de manière à permettre le remplacement de la couche de synchronisation et la comparaison des deux approches décrites sur la même interface utilisateur.

Le client maintient une réplique locale du document et convertit chaque modification de l'utilisateur en une opération sur le type de données répliqué. Les opérations sont envoyées au serveur par lots, afin de ne pas saturer le réseau lors d'une saisie rapide. Le listing suivant présente le traitement simplifié d'un lot entrant sur le serveur :

```
def traiter_lot(document, lot):
    for operation in lot.operations:
        if operation.id in document.appliquees:
            continue
        document.appliquer(operation)
        document.appliquees.add(operation.id)
    diffuser_aux_autres(document.id, lot)
```

Comme les opérations commutent, le serveur n'a besoin ni d'ordonner ni de transformer les lots. Il lui suffit de garantir qu'aucune opération n'est exécutée deux fois et que chaque lot parvient à tous les clients connectés.

3.1 . Stockage de l'historique

Le serveur n'enregistre pas seulement le texte final, mais tout le journal des opérations. À partir du journal, il est possible de restaurer n'importe quelle version antérieure du document et de reconstituer l'état de la réplique d'un client resté longtemps déconnecté. Cette approche correspond à une architecture fondée sur un enregistrement immuable des événements, telle que la décrit Kleppmann [3]. Pour que le journal ne croisse pas indéfiniment, le serveur le compacte à intervalles réguliers sous forme d'instantanés.

4 - Évaluation

Nous avons comparé le prototype à une implémentation de référence de la transformation opérationnelle sur un ensemble d'enregistrements de sessions d'édition réelles. Pour chaque session, nous avons mesuré la latence, c'est-à-dire le temps écoulé entre une modification chez son auteur et son affichage chez les autres participants, et nous avons calculé la latence moyenne de la session selon (4.1) :

$$\bar{t} = \frac{1}{n} \sum_{i=1}^n (t_i^{\text{réception}} - t_i^{\text{modification}}) \quad (4.1)$$

où n est le nombre d'opérations dans la session. Les valeurs mesurées pour différents nombres d'utilisateurs travaillant simultanément sont résumées dans le Table 4.1.

Table 4.1 – Latence moyenne et taille des données stockées selon le nombre d'utilisateurs

Utilisateurs	Latence OT (ms)	Latence CRDT (ms)	Données (ko)
2	38	35	12
5	52	41	19
10	87	48	31
20	164	63	58

À mesure que le nombre d'utilisateurs augmente, la latence de la transformation opérationnelle croît nettement plus vite, car le serveur doit transformer chaque opération par rapport à toutes les opérations concurrentes. Le prix de la latence plus faible des CRDT est un volume plus important de données stockées, puisque chaque caractère porte son identifiant. La répartition des modifications dans le temps, sur laquelle repose le test de charge, est présentée à la Figure 4.1.

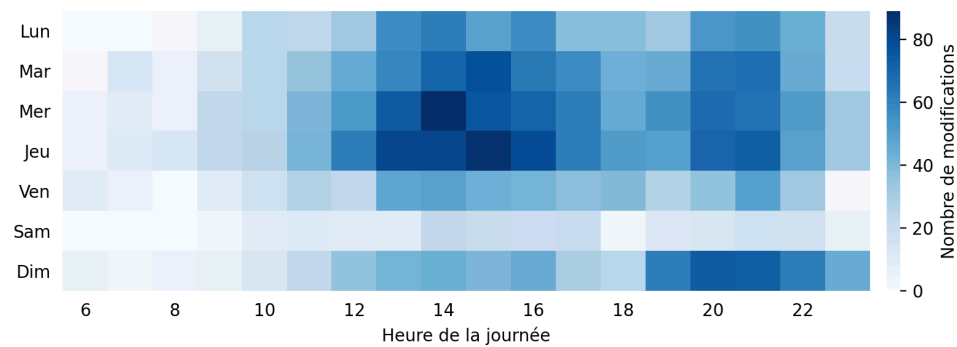


Figure 4.1 – Nombre de modifications par jour de la semaine et par heure dans les sessions étudiées

4.1 . Discussion des résultats

Les résultats confirment que, pour les petites équipes, les deux approches se valent et que c'est plutôt la simplicité de mise en œuvre qui est déterminante. Avec un plus grand nombre d'utilisateurs simultanés, les CRDT présentent un net avantage tant en latence qu'en robustesse face aux coupures de connexion. Des entretiens complémentaires avec les utilisateurs, menés selon les principes de la recherche qualitative [4], ont montré que les utilisateurs ne perçoivent pas une différence de latence inférieure à cent millisecondes, mais qu'ils considèrent la perte du texte en cours de rédaction après une coupure de connexion comme un problème majeur.

Conclusion

Ce travail a comparé la transformation opérationnelle et les types de données répliqués sans conflit en tant que deux approches de la synchronisation des documents lors de l'édition collaborative en temps réel. Sur un prototype d'éditeur web, nous avons montré que les CRDT permettent un serveur plus simple, sans ordonnancement central des opérations, et offrent une latence plus faible lorsque le nombre d'utilisateurs est plus élevé.

Le principal inconvénient des CRDT reste le volume plus important de données stockées. La prochaine étape consistera donc à vérifier comment le compactage régulier du journal en instantanés influe sur la taille des données et comment le prototype se comporte lors d'un fonctionnement prolongé avec des dizaines de milliers de documents.

Bibliographie

- [1] C. A. Ellis et S. J. Gibbs, « Concurrency control in groupware systems », in *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, New York : ACM, 1989, p. 399-407. doi : [10.1145/67544.66963](https://doi.org/10.1145/67544.66963).
- [2] M. Shapiro, N. Preguiça, C. Baquero, et M. Zawirski, « Conflict-free replicated data types », in *Stabilization, Safety, and Security of Distributed Systems*, Berlin : Springer, 2011, p. 386-400. doi : [10.1007/978-3-642-24550-3_29](https://doi.org/10.1007/978-3-642-24550-3_29).
- [3] M. Kleppmann, *Designing data-intensive applications*. Sebastopol : O'Reilly Media, 2017.
- [4] J. Hendl, *Kvalitativní výzkum : základní metody a aplikace*. Praha : Portál, 2005.