Ré-identification sans coordination dans les types de données répliquées sans conflits

Matthieu Nicolas (matthieu.nicolas@loria.fr)

20 décembre 2022

Rapporteurs :	Hanifa Boucheneb	Professeure, Polytechnique Montréal
	Davide Frey	Chargé de recherche, HdR, Inria Rennes Bretagne-Atlantique
Examinateurs :	Hala Skaf-Molli	Maîtresse de conférences, HdR, Nantes Université, LS2N
	Stephan Merz	Directeur de Recherche, Inria Nancy - Grand Est
Encadrants :	Olivier Perrin	Professeur des Universités, Université de Lorraine, LORIA
	Gérald Oster	Maître de conférences, Université de Lorraine, LORIA

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Inria

cnrs
dépasser les frontières

Applications collaboratives

TODO : Voir comment représenter une appli collaboratice à ce stade

- Un système collaboratif est un système supportant ses utilisateur-rices dans leurs processus de collaboration pour la réalisation de tâches.

Démocratisation des applications collaboratives

TODO : Voir comment illustrer ce point. Screenshots et nombre d'utilisateurs en-dessous ?

Avantages d'une architecture basée sur le cloud…

TODO : Représenter collaboration via appli basée sur le cloud

- Disponibilité : Répond aux utilisateur-rices
- Tolérance aux pannes : Fonctionne malgré pannes
- Capacité de passage à l'échelle : Supporte activité massive

… et ses limites

TODO : Illustrer chacune des propriétés

- Confidentialité
- Souveraineté
- Pérennité
- Résistance à la censure

Pouvons-nous concevoir des applications collaboratives satisfaisant l'ensemble de ces propriétés ?

Applications collaboratives pair-à-pair [1]

TODO : Illustrer une appli P2P

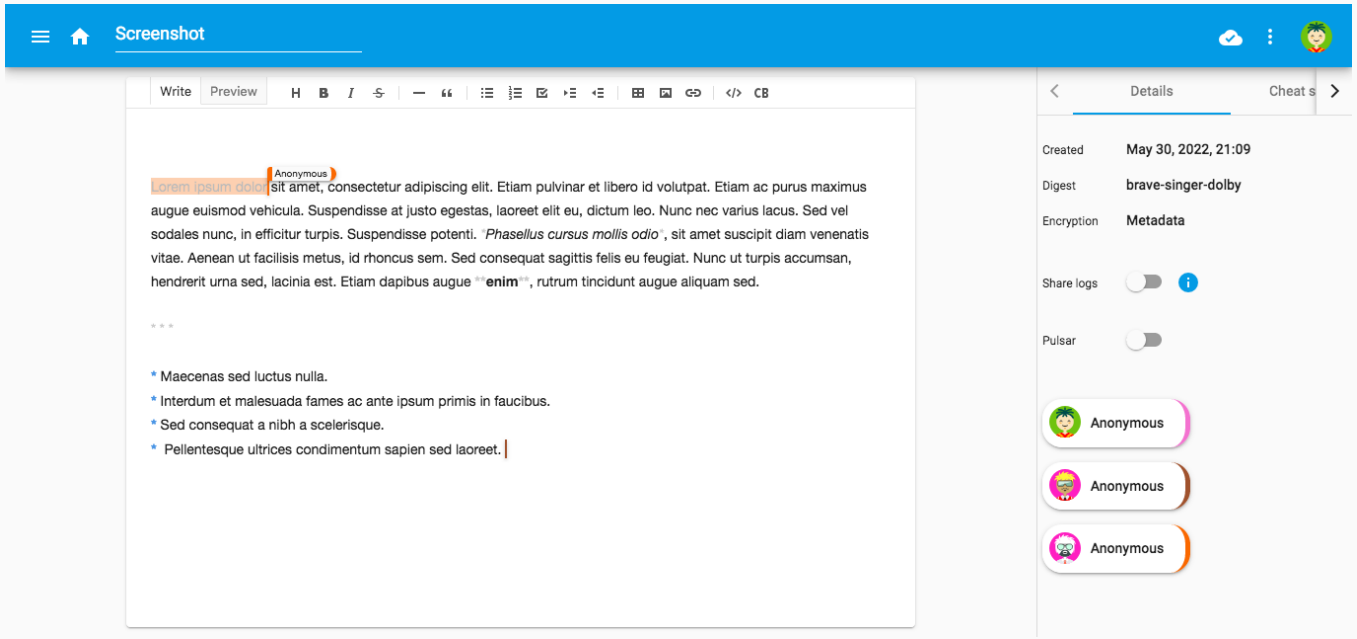
Problématiques

En l'absence d'autorités centrales, comment

- résoudre les conflits de modifications ?
- authentifier les utilisateur-rices ?
- sécuriser les communications ?

[1]. Kleppmann et al., « Local-First Software : You Own Your Data, in Spite of the Cloud ».

MUTE *



- Éditeur de texte collaboratif P2P temps réel chiffré de bout en bout
- Permet à l'équipe d'étudier et contribuer sur les problématiques des applications Local-First Software (LFS)

*. Disponible à : `https://mutehost.loria.fr`

Mes contributions

TODO : À retravailler pour faire apparaître les problématiques ?
TODO : Ajouter comparaison des modèles de sync et approches pour CRDTs pour le type Séquence ?

- Conception d'un nouveau Conflict-free Replicated Data Type (CRDT) pour le type Séquence
- Implémentation et intégration de CRDTs dans MUTE

Type de données *Séquence*

TODO : Représenter une séquence, puis une exécution provoquant un conflit

- Représente une collection ordonnée et de taille dynamique d'éléments
- Deux opérations de modifications, *insert* et *remove*
- Modifications ne commutent généralement pas

Un mécanisme de résolution de conflits est nécessaire pour résoudre les divergences

Conflict-free Replicated Data Types (CRDTs) [2]

- Nouvelles spécifications des types de données
- Types de données répliquées
  - Permettent modifications sans coordination
  - Garantissent la convergence forte

Convergence forte

Ensemble des noeuds ayant intégrés le même ensemble de modifications obtient des états équivalents, sans nécessiter d'actions ou messages supplémentaires

[2]. Shapiro et al., « Conflict-Free Replicated Data Types ».

CRDTs et théorie des treillis [3]

- Reposent sur la théorie des treillis

TODO : Représenter un sup-demi-treillis

- États ordonnés par relation $\leq$, avec état minimal $\bot$
- Modification d'un état génère état supérieur ou égal d'après $\leq$
- Existe fonction qui fusionne deux états et retourne leur borne supérieure

[3]. Davey et al., *Introduction to Lattices and Order*.

La fonction de fusion d’états est un mécanisme de résolution de conflits automatique

Caractéristiques d'un CRDT

Ce qui définit un CRDT

- Sémantique de son mécanisme de résolution de conflits
- Modèle de synchronisation utilisé

Se focalise sur

- Spécification forte des séquences répliquées avec entrelacements [4]
- Synchronisation par opérations

[4]. Attiya et al., « Specification and space complexity of collaborative text editing ».

LogootSplit [5]

- CRDT pour le type Séquence
- Appartient à l'approche à identifiants densément ordonnés

[5]. André et al., « Supporting Adaptable Granularity of Changes for Massive-Scale Collaborative Editing ».

Approche à identifiants densément ordonnés

- Assigne un identifiant de position à chaque élément de la séquence
- Utilise les identifiants des éléments pour les ordonner les uns par rapport aux autres

Propriétés des identifiants de position [6]

- Unique
- Immuable
- Ordonnable par une relation d'ordre strict total $<_{id}$
- Appartenant à un espace dense

[6]. Preguica et al., « A Commutative Replicated Data Type for Cooperative Editing ».

Identifiant LogootSplit

- Composé d'un ou plusieurs tuples $pos_{offset}^{nodeId\ nodeSeq}$
- TODO : Ajouter animations pour expliciter le rôle de chaque composant

Exemple :

$$i_0^{A1} r_3^{B1} \cdots m_0^{A2}$$

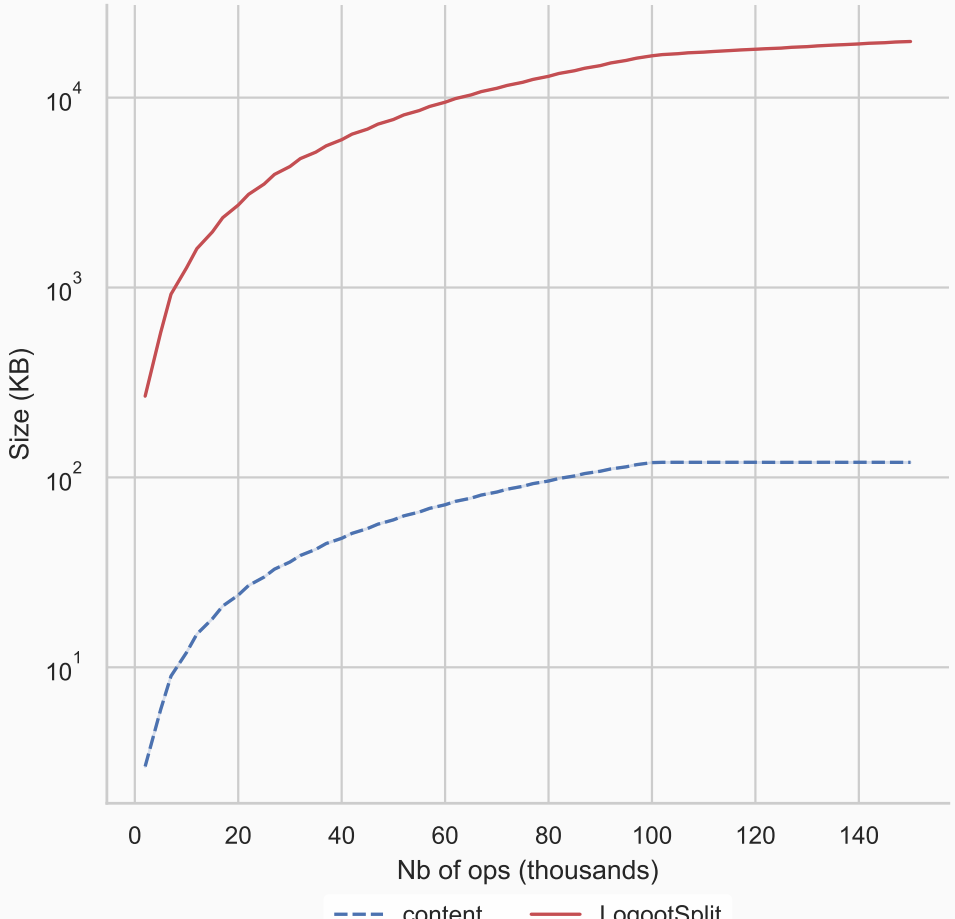
???

TODO : Reprendre exemple de divergence, mais avec LogootSplit pour montrer convergence

Limites de LogootSplit

- Croissance non-bornée de la taille des identifiants
- Fragmentation en blocs courts

Taille du contenu comparé à la taille de la séquence LogootSplit



1% contenu, 99% métadonnées

Mitigation du surcoût des CRDTs pour le type Séquence

L'approche core-nebula [7]

- Ré-assigne des identifiants courts aux éléments, c.-à-d. les *renomme*
- Nécessite de transformer les opérations *insert* et *remove* concurrentes...
- ...et consensus pour effectuer le renommage

Inadaptée aux systèmes P2P sujets au churn

[7]. Zawirski et al., « Asynchronous rebalancing of a replicated tree ».

Pouvons-nous proposer un mécanisme de réduction du surcoût des CRDTs pour le type Séquence adapté aux applications LFS, c.-à-d. sans coordination synchrone ?

RenamableLogootSplit

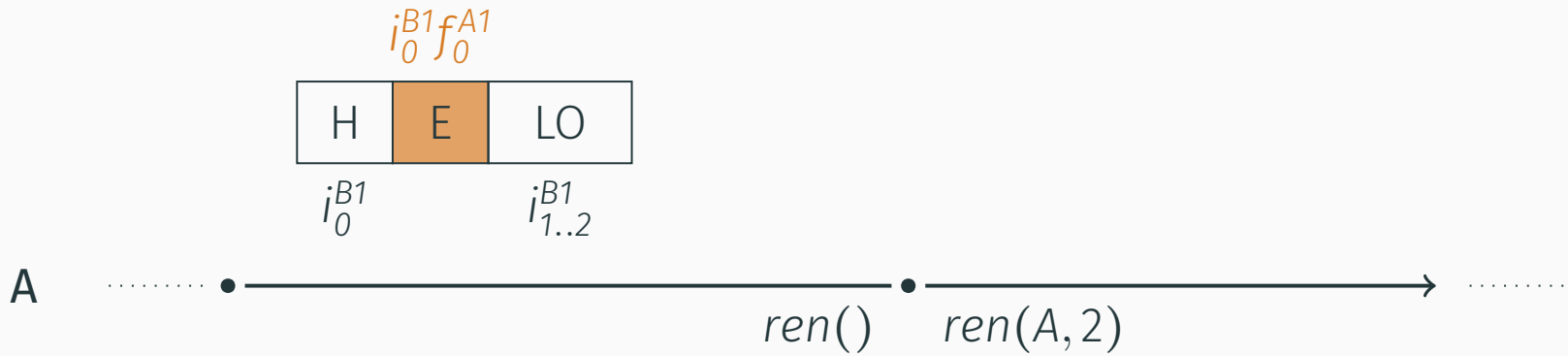
Contribution : RenamableLogootSplit

- CRDT pour le type Séquence qui incorpore un mécanisme de renommage
- Prend la forme d'une nouvelle opération : *rename*

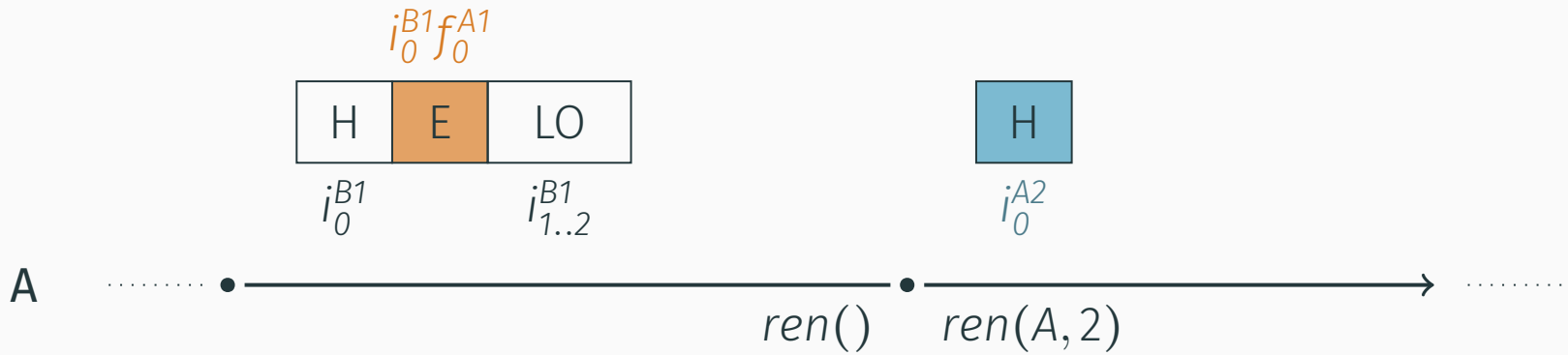
Propriétés de l'opération *rename*

- Est déterministe
- Préserve l'intention des utilisateur-rices
- Préserve la séquence, c.-à-d. unicité et ordre de ses identifiants
- Commute avec les opérations *insert*, *remove* mais aussi *rename* concurrentes

Opération *rename*

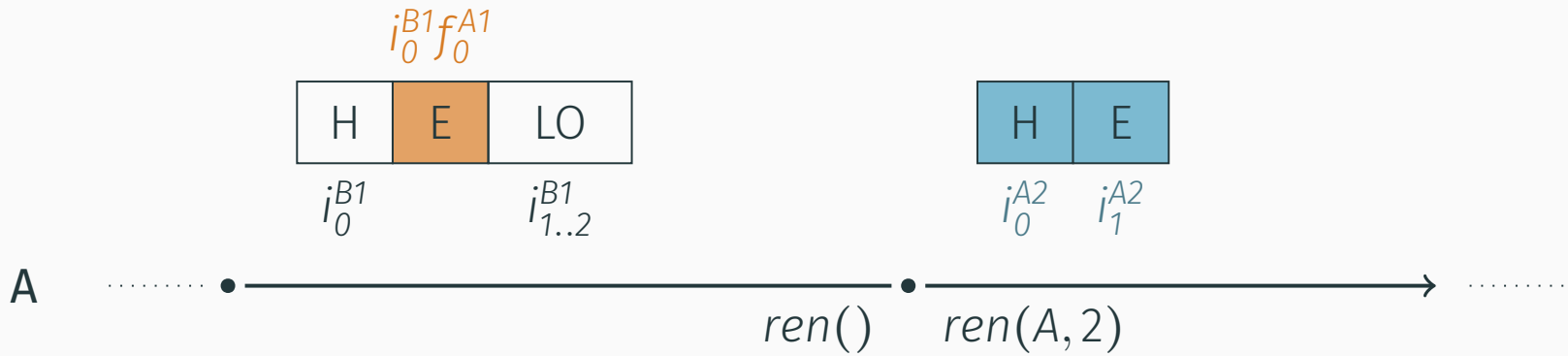


Opération *rename*



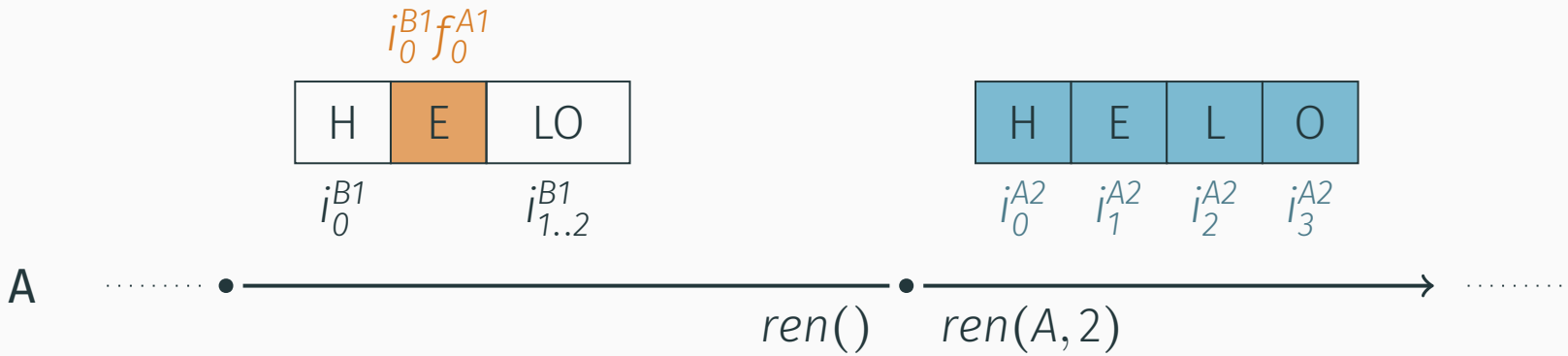
- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$

Opération *rename*



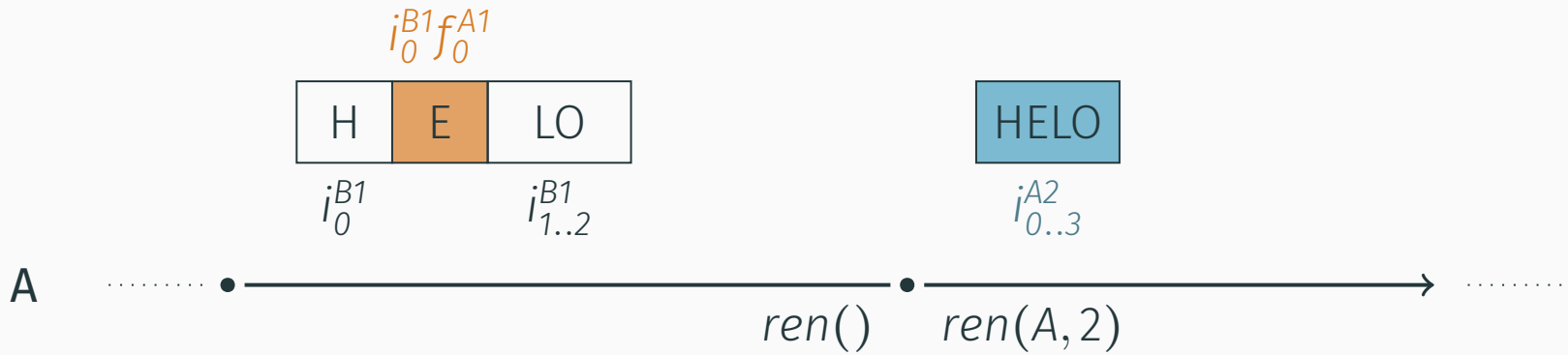
- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants : i_1^{A2}, i_2^{A2}...

Opération *rename*



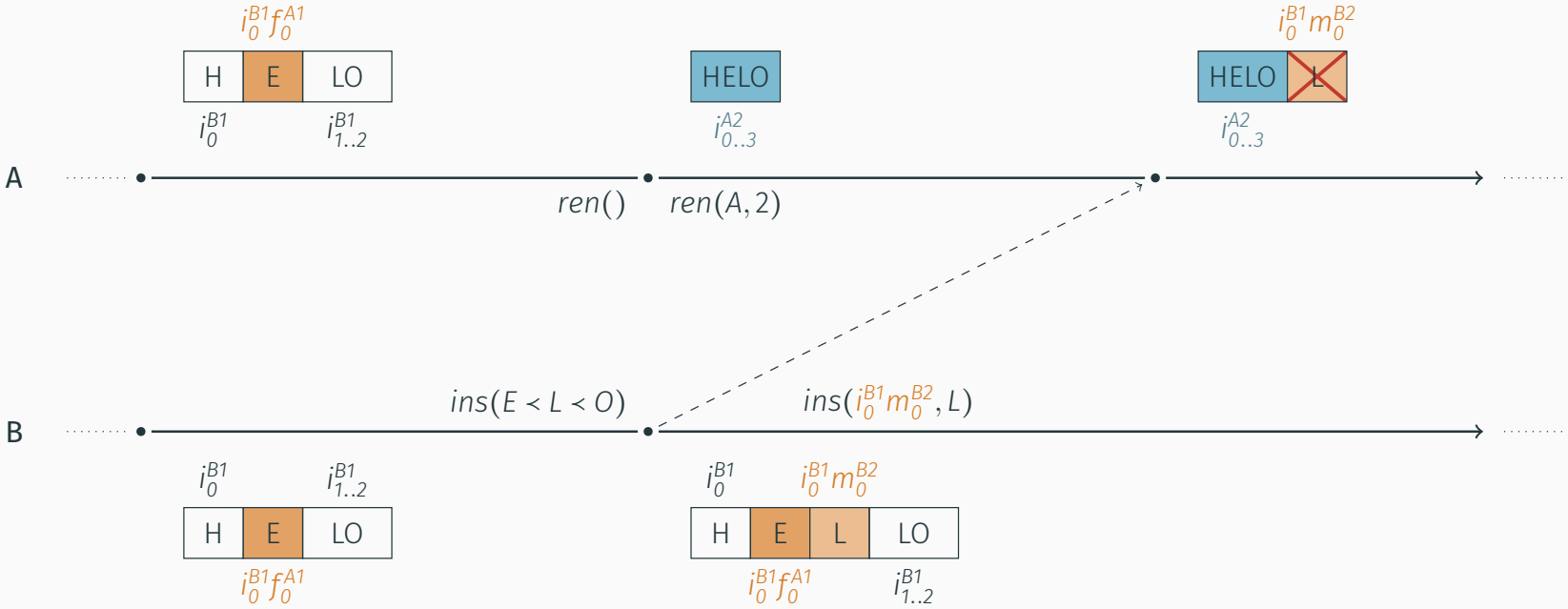
- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants : i_1^{A2}, i_2^{A2}...

Opération *rename*



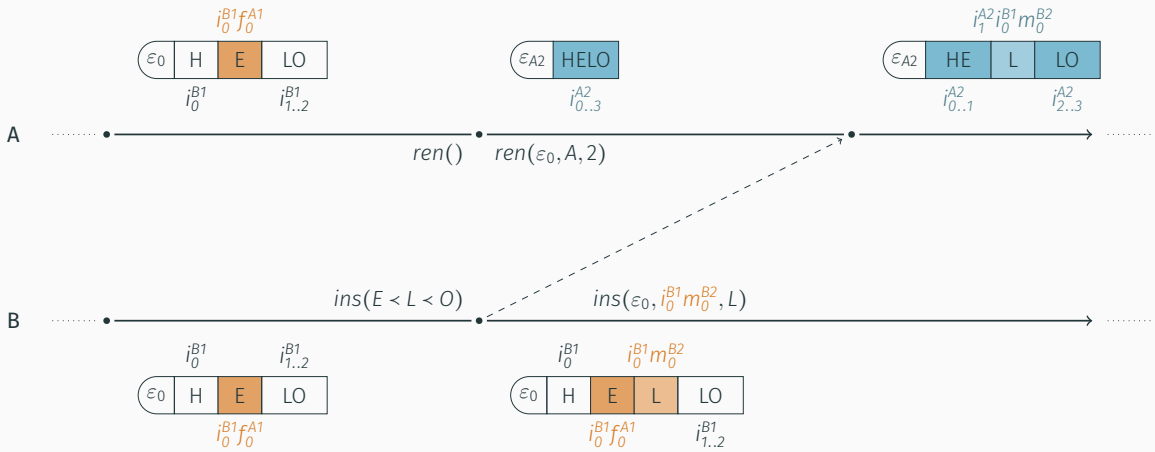
- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants : i_1^{A2}, i_2^{A2}...

Intégration des opérations *insert* et *remove* concurrentes



- Peuvent générer opérations concurrentes aux opérations *rename*
- Produisent anomalies si intégrées telles qu'elles

Correction de l'intégration des opérations concurrentes



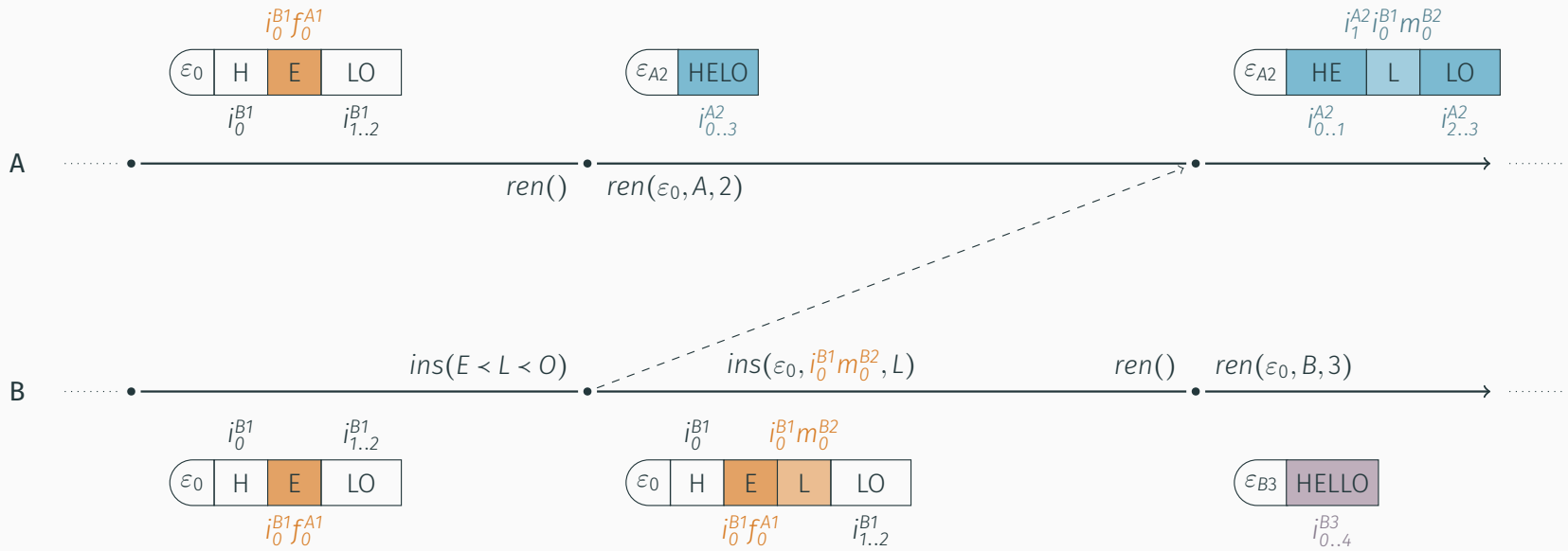
- Ajout d'un système d'époque pour identifier les opérations concurrentes à une opération *rename*
- Transformation avant intégration des opérations concurrentes

Exemple avec $i_0^{B1} m_0^{B2}$

- Trouver son prédecesseur à l'époque d'origine ε_0 : $i_0^{B1} f_0^{A1}$
- Trouver son équivalent à l'époque destination ε_{A1} : i_1^{A2}
- Concaténer ce dernier à l'identifiant pour obtenir son équivalent à ε_{A1} : $i_1^{A2} i_0^{B1} m_0^{B2}$

Et en cas d’opérations *rename* concurrentes ?

Opérations *rename* concurrentes



- Apparition d'une divergence

Besoin d'un mécanisme de résolution de conflits pour faire converger les noeuds

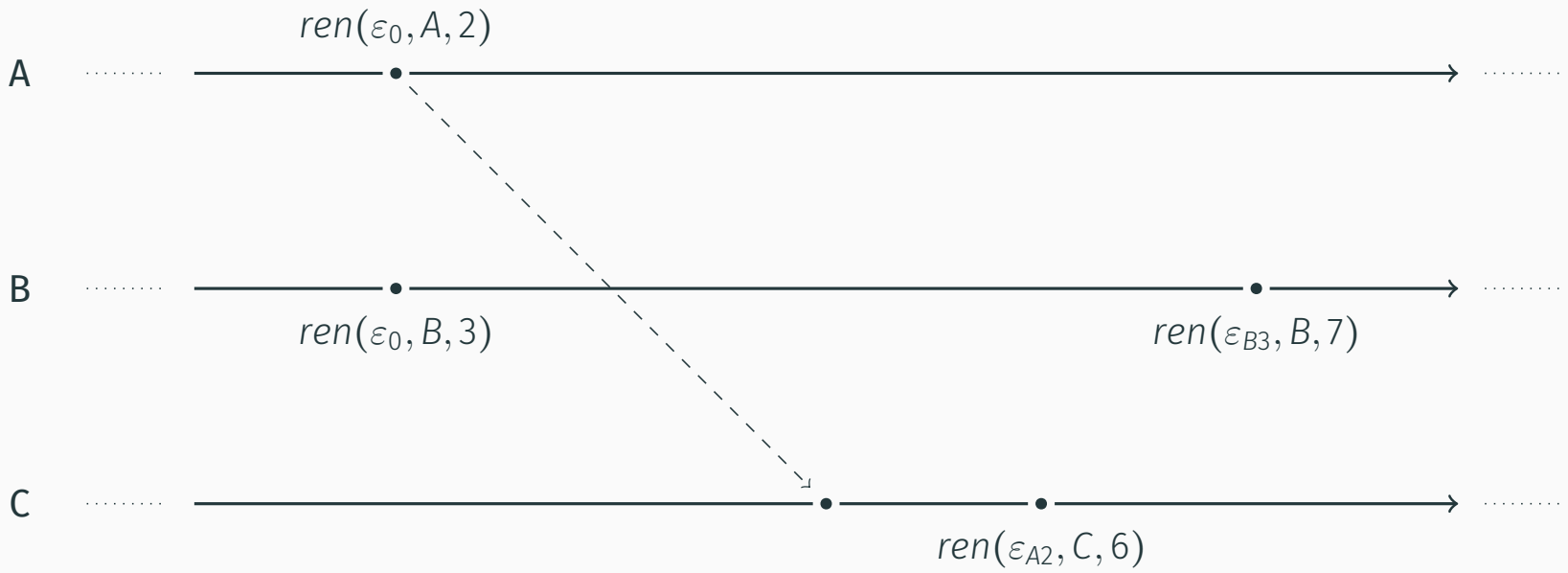
Résolution de conflits entre opérations *rename* concurrentes

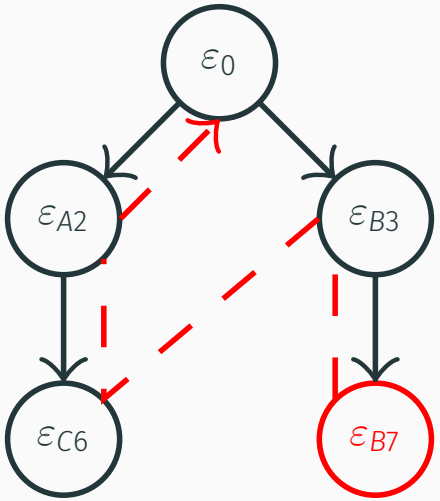
- Opérations *rename* sont des opérations systèmes, c.-à-d. pas d'intention utilisateur
- Peut ne pas appliquer les effets de certaines d'entre elles

Intuition

- Choisir une époque comme époque cible
- Calculer chemin entre époque courante et époque cible, et notamment leur Plus Proche Ancêtre Commun (PPAC)
- Annuler l'effet des opérations *rename* de l'époque courante au PPAC
- Appliquer l'effet des opérations *rename* du PPAC à l'époque cible

Choisir une époque comme époque cible





- Doit définir relation *priority*, notée $<_\varepsilon$, ordre strict total sur les époques
- Utilise ordre lexicographique sur chemins des époques dans l'arbre

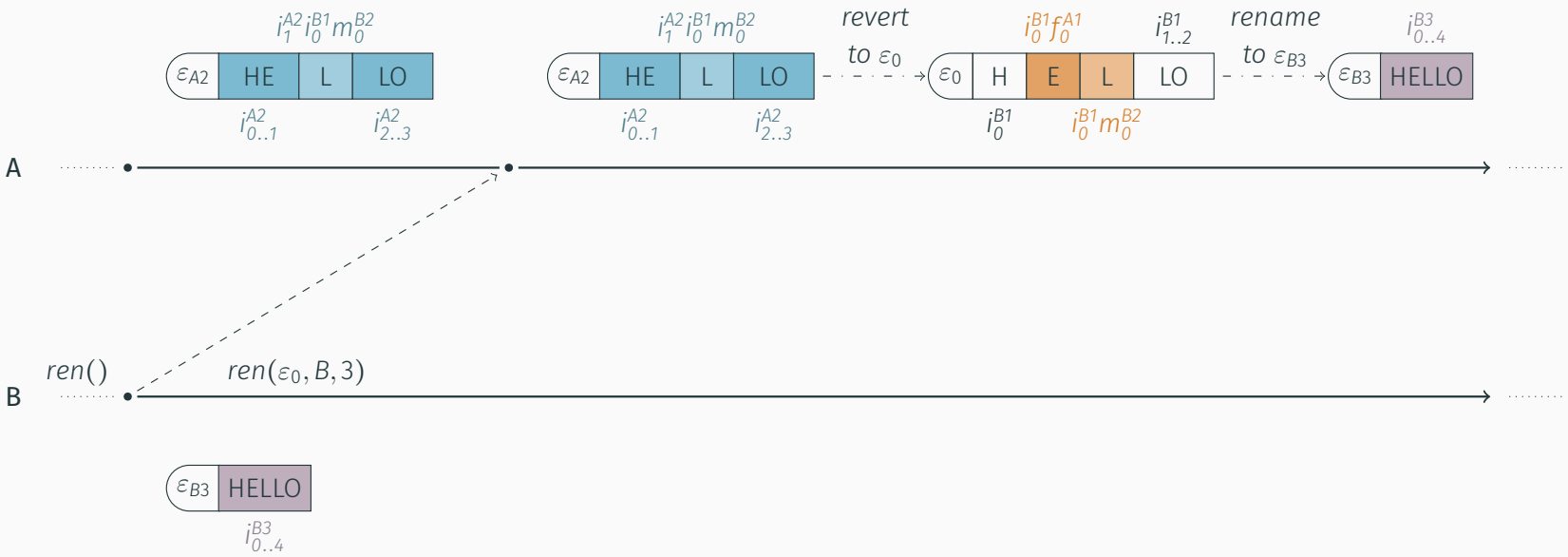
Annuler l'effet d'une opération *rename*

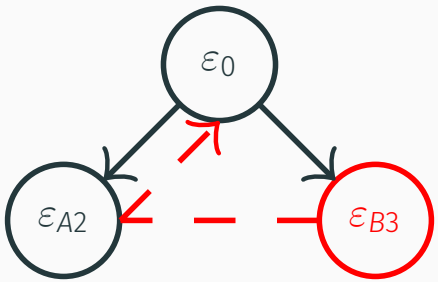
- Propose un nouvel algorithme, *revertRenameId*
- Permet d'obtenir l'identifiant correspondant à l'époque parente

Intuition

1. *id* fait partie des identifiants renommés : doit retourner son ancienne valeur
2. *id* a potentiellement été inséré en concurrence : doit restaurer son ancienne valeur
3. *id* a été inséré après le renommage : doit retourner une valeur qui préserve l'ordre

Opérations *rename* concurrentes





- TODO : Illustrer choix de l'époque cible, cas 1 et cas 2

RenamableLogootSplit

Validation

Objectifs

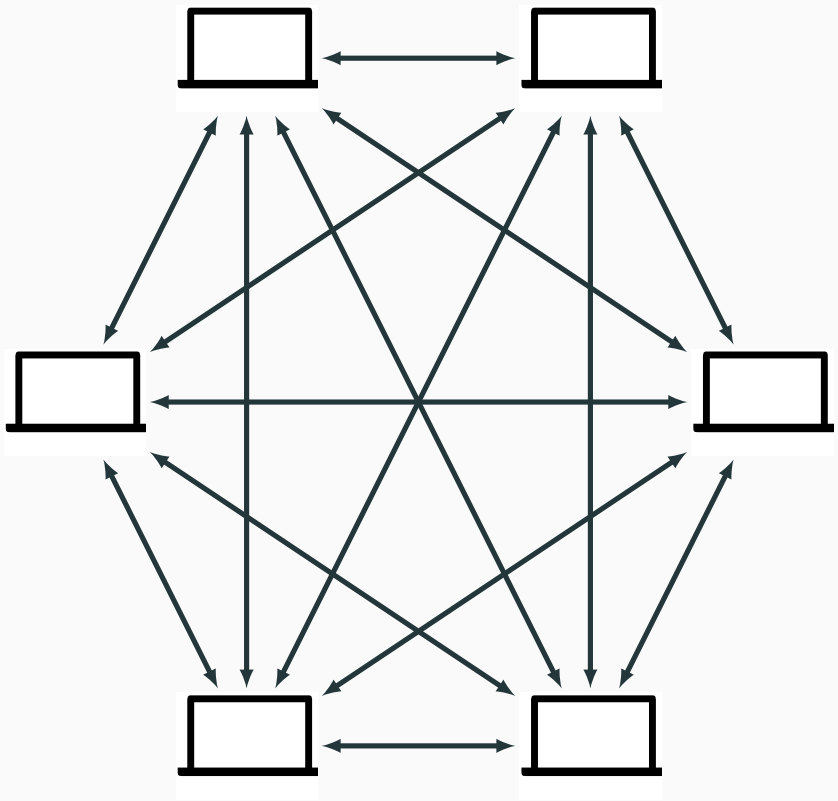
- Montrer convergence des noeuds
- Montrer que mécanisme de renommage améliore performances de la séquence répliquée (mémoire, calculs, bande-passante)

Conduction d'une évaluation expérimentale

Absence d’un jeu de données de sessions d’édition collaborative

Mise en place de simulations pour générer un jeu de données

Simulation - Architecture



- 10 noeuds éditent collaborativement un document
- Utilisent soit LogootSplit (LS), soit RenamableLogootSplit (RLS)
- Topologie réseau entièrement maillée
- Ne considère pas pannes ou pertes de message

Simulation - Modifications

- Phase 1 (génération du contenu) : Beaucoup d'insertions, quelques suppressions (80/20%)
- Phase 2 (édition) : Équilibre insertions/suppressions (50/50%)
- Noeuds passent à la phase 2 quand document atteint taille donnée (15 pages - 60k caractères)

- Noeuds terminent quand ensemble des noeuds a effectué nombre donné de modifications (10k)...
- ...et intégré celles des autres (150k au total)

Simulation - Mécanisme de renommage

- Noeuds désignés comme *noeuds de renommage* (1 à 4)
- Noeuds de renommage effectue un renommage à toutes les 7.5k/30k opérations qu'ils intègrent (5/20 opérations *rename* par noeud de renommage)
- Opérations *rename* générées à un point donné sont concurrentes

Simulation - Sorties

- Instantané de l'état de chaque noeud à différents points de la simulation (2.5k/10k opérations et état final)
- Journal des opérations de chaque noeud

Permet de conduire évaluations sur ces données *

*. Code des simulations et benchmarks :
`https://github.com/coast-team/mute-bot-random`

RenamableLogootSplit

Résultats

Convergence

Intuition

Comparer l'état final des différents noeuds d'une session pour confirmer l'absence de divergence

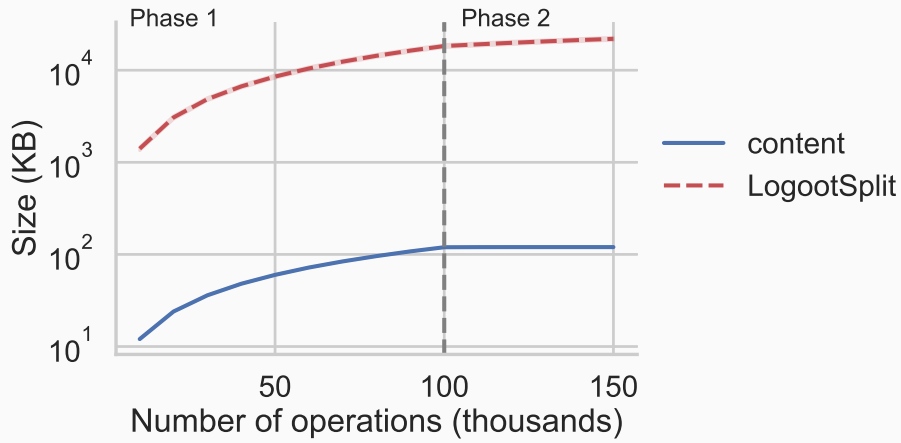
- Ensemble des noeuds convergent
- Un résultat empirique, pas une preuve...
- ...mais un premier pas vers la validation de RLS

Surcoût en métadonnées - 1 noeud de renommage

Intuition

Mesurer évolution de la taille de la structure de données à partir des instantanés des sessions avec 1 seul noeud de renommage

TODO : Regénérer figure avec légende en bas

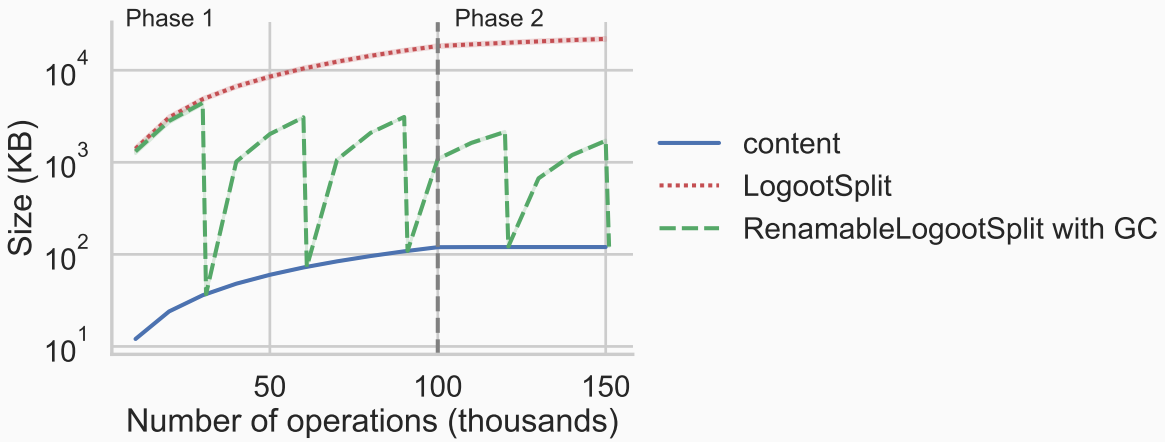


Surcoût en métadonnées - 1 noeud de renommage

Intuition

Mesurer évolution de la taille de la structure de données à partir des instantanés des sessions avec 1 seul noeud de renommage

TODO : Regénérer figure avec légende en bas



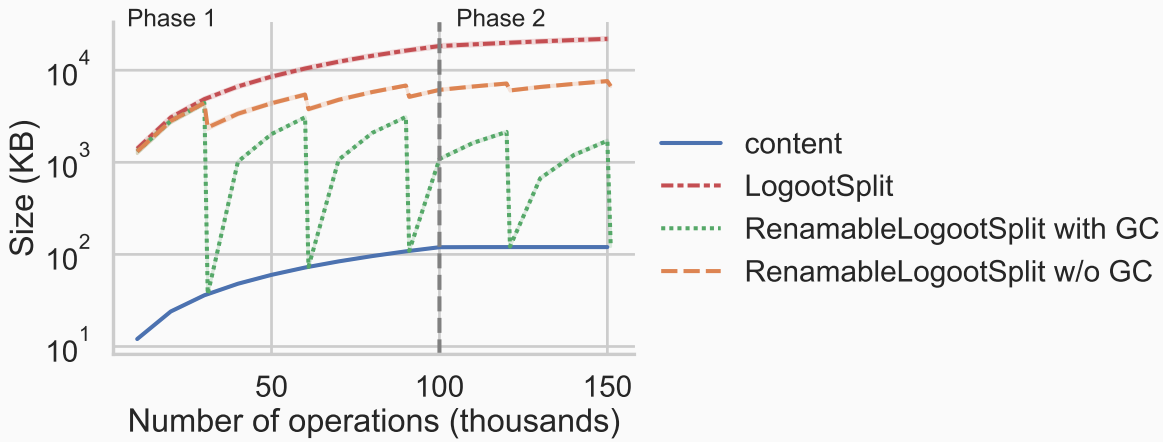
- Opération *rename* réinitialise le surcoût du CRDT, si GC de l'entièreté des métadonnées du mécanisme de renommage

Surcoût en métadonnées - 1 noeud de renommage

Intuition

Mesurer évolution de la taille de la structure de données à partir des instantanés des sessions avec 1 seul noeud de renommage

TODO : Regénérer figure avec légende en bas



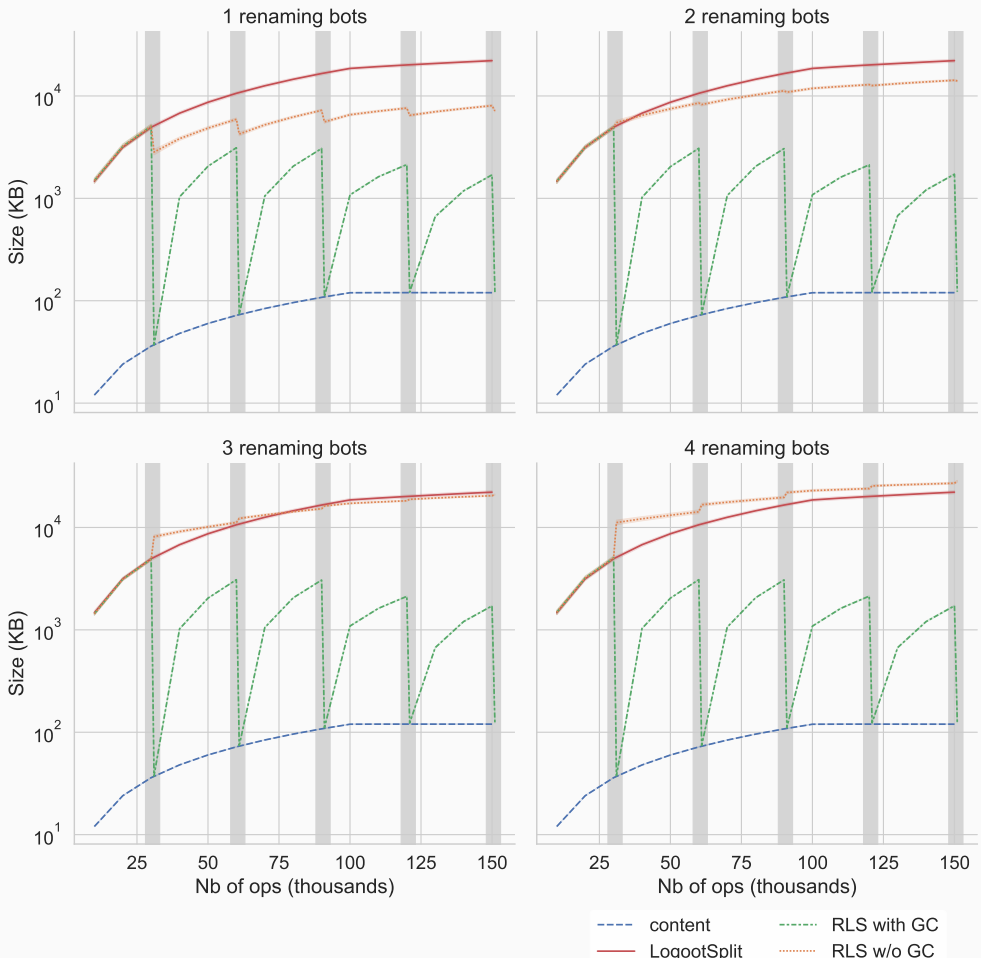
- Opération *rename* réinitialise le surcoût du CRDT, si GC de l'entièreté des métadonnées du mécanisme de renommage
- Opération *rename* réduit de 66% le surcoût du CRDT sinon

Surcoût en métadonnées - 1 à 4 noeuds de renommage

Intuition

Mesurer évolution de la taille de la structure de données en fonction du nombre de noeuds de renommage

TODO : Mettre en ligne les 4 plots



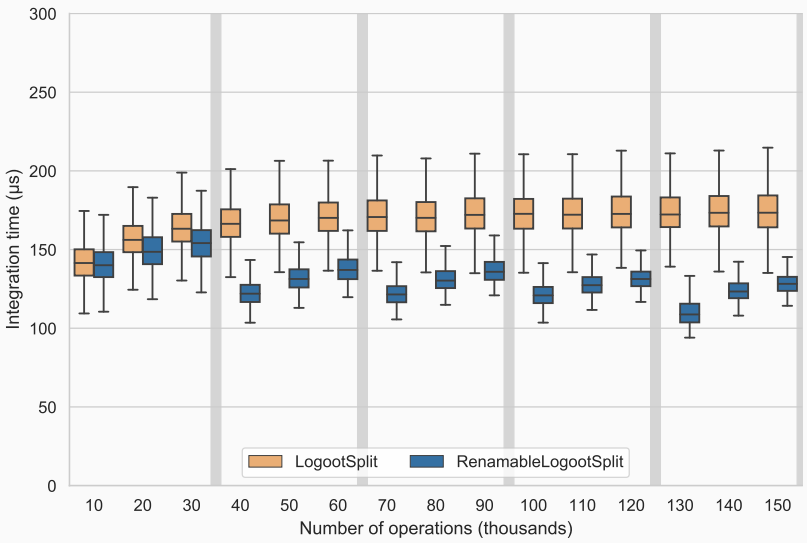
- Aucun impact si GC
- Surcoût de chaque opération *rename* s'additionne sinon

Surcoût en calculs - Opérations *insert* et *remove*

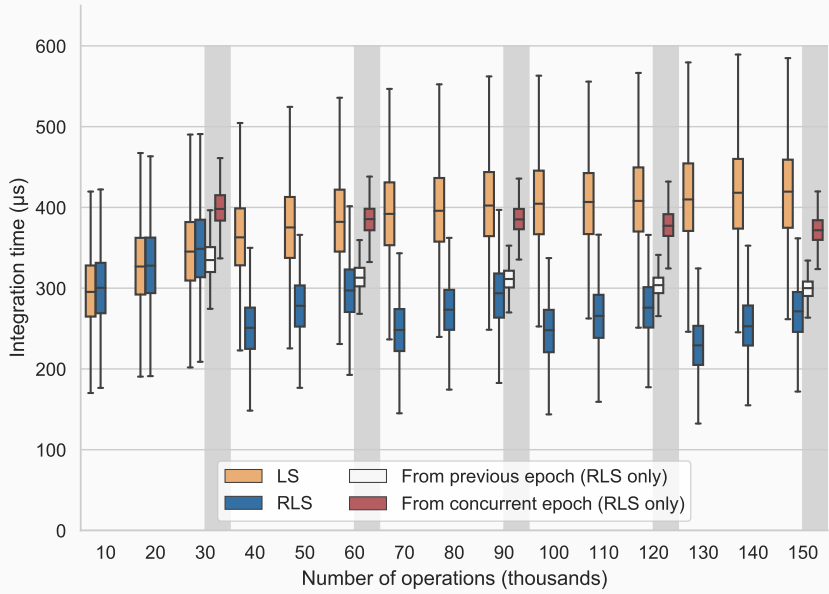
Intuition

Mesurer temps d'intégration local et distant d'opérations *insert* à différents stades de la collaboration

TODO : Animer l'apparition au fur et à mesure des différentes stats ?



(a) Modifications locales



(b) Modifications distantes

- Opérations *rename* réduisent temps d'intégration des

Surcoûts en calculs - Opérations *rename*

Intuition

Mesurer temps d'intégration local et distant d'opérations *rename* à différents stades de la collaboration

Paramètres		Temps d'intégration (ms)				
Type	Nb Ops (k)	Moyenne	Médiane	IQR	1er Percent.	99ème Percent.
Locale	30	41.8	38.7	5.66	37.3	71.7
	90	119	119	2.17	116	124
	150	158	158	3.71	153	164
Distante directe	30	481	477	15.2	454	537
	90	1491	1482	58.8	1396	1658
	150	1694	1676	60.6	1591	1853
Cc. int. époque plus prioritaire	30	644	644	16.6	620	683
	90	1998	1994	46.6	1906	2112
	150	2242	2234	63.5	2139	2351

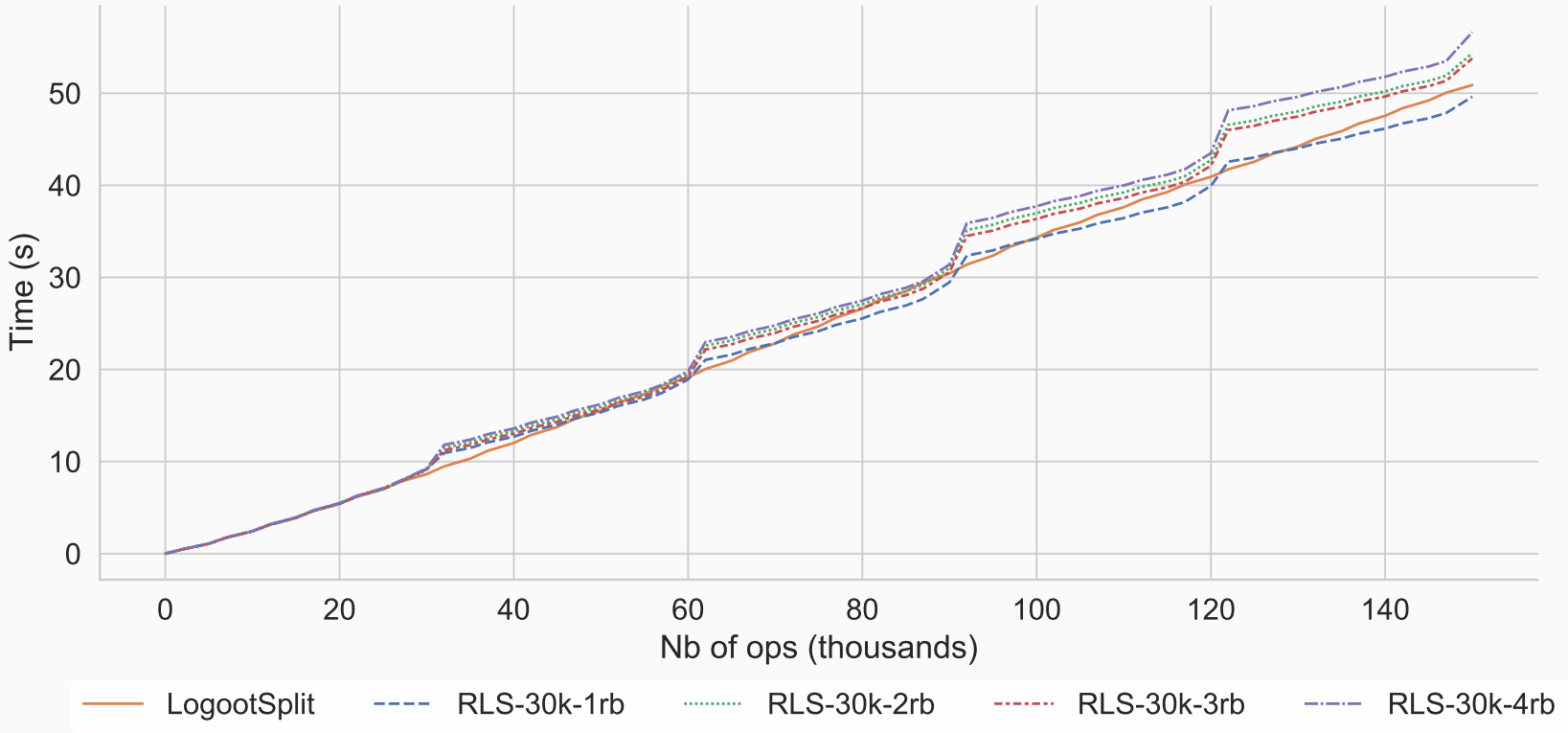
- Détectable par utilisateur-rices
- Nécessaire d'améliorer temps d'intégration distant

Surcoût en calculs - Vue globale

Intuition

Mesurer temps pour intégrer l'entièreté du journal d'opérations d'une collaboration en fonction du nombre de noeuds de renommage

TODO : Afficher résultats qu'à partir de 75k opés



RenamableLogootSplit

Conclusion

Conclusion

Contribution

- Définition et validation mécanisme de renommage pour CRDTs pour Séquence à identifiants densément ordonnés, compatible avec systèmes pair-à-pair

Limites

- Surcoût fonction du nombre d'opérations *rename* concurrentes
- Stabilité causale [8] requise pour supprimer les métadonnées

Perspectives

- Stratégies de génération des opérations *rename*
- Relations *priority* $<_\varepsilon$ réduisant calculs à l'échelle du système
- Preuve de correction de RLS

[8]. Baquero et al., *Pure Operation-Based Replicated Data Types*.

Conclusion générale & Perspectives

Conclusion

Contributions

- Mécanisme de renommage pour CRDTs pour le type Séquence à identifiants densément ordonnés
- Implémentation de RenamableLogootSplit et de ses dépendances (protocole d'appartenance au réseau, couche de livraison) dans MUTE
- Comparaison des différents modèles de synchronisation pour CRDTs, ainsi que
- Comparaison des différentes approches pour CRDTs pour le type Séquence

Perspectives

RenamableLogootSplit

- Relations *priority* $<_\varepsilon$ réduisant calculs à l'échelle du système

CRDTs

- Framework pour conception de CRDTs synchronisés par opérations
- Comparaison des différents modèles de synchronisation pour CRDTs

Ouverture : comparaison modèles synchronisation

	Sync. par états	Sync. par opérations	Sync. par diff. d'états
Forme un sup-demi-treillis	✓	✓	✓
Intègre modifications par fusion d'états	✓	✗	✓
Intègre modifications par élts irréductibles	✗	✓	✓
Résiste nativ. aux défaillances réseau	✓	✗	✓
Adapté pour systèmes temps réel	✗	✓	✓
Offre nativ. modèle de cohérence causale	✓	✗	✗

- Synchronisation par différences offre meilleur des mondes…
- …y a-t-il encore un intérêt aux autres modèles, e.g. pour composition ou sécurité ?

Merci de votre attention, avez-vous des questions ?

Publications

- **Article de position** à Middleware 2018 - 19th ACM/IFIP International Middleware Conference (Doctoral Symposium), Dec 2018, Rennes, France.
- **Article d'atelier** avec Gérald Oster et Olivier Perrin à PaPoC 2020 - 7th Workshop on Principles and Practice of Consistency for Distributed Data, Apr 2020, Heraklion / Virtual, Greece.
- **Article de revue** avec Gérald Oster et Olivier Perrin dans IEEE Transactions on Parallel and Distributed Systems, Institute of Electrical and Electronics Engineers, 2022, 33 (12), pp.3870-3885.

Benchmarks

- Node.js, version 13.1.0, avec option `jitless`
- Machiné équipée d'un Intel Xeon CPU E5-1620 (10MB Cache, 3.50 GHz), de 16GB de RAM et utilisant Fedora 31
- Taille des documents obtenus en utilisant notre fork de *object-sizeof*[*]
- Mesures de temps avec `process.hrtime.bigint()`

*. `https://www.npmjs.com/package/object-sizeof`