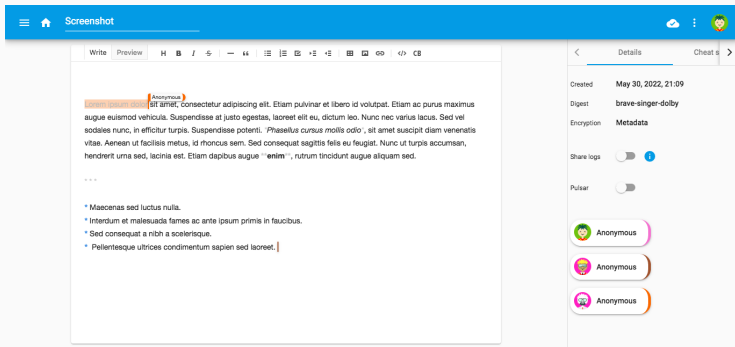


Ré-identification sans coordination dans les types de données répliquées sans conflits

Matthieu Nicolas (matthieu.nicolas@loria.fr)

20 décembre 2022

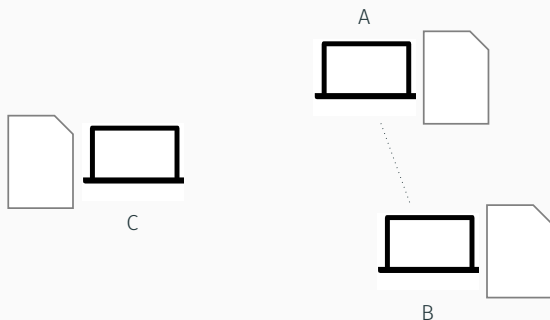
<i>Rapporteurs :</i>	Hanifa Boucheneb Davide Frey	Professeure, Polytechnique Montréal Chargé de recherche, HdR, Inria Rennes Bretagne-Atlantique
<i>Examineurs :</i>	Hala Skaf-Molli Stephan Merz	Maîtresse de conférences, HdR, Nantes Université, LS2N Directeur de Recherche, Inria Nancy - Grand Est
<i>Encadrants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



- Application pair-à-pair
- Permet à groupes de rédiger collaborativement documents texte
- Garantit confidentialité & souveraineté de ses données

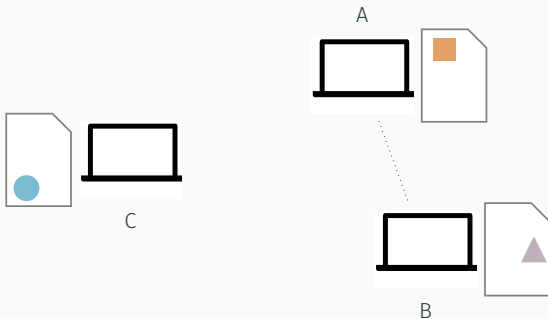
*. Disponible à : <https://mutehost.loria.fr>

Réplication dans applications collaboratives pair-à-pair



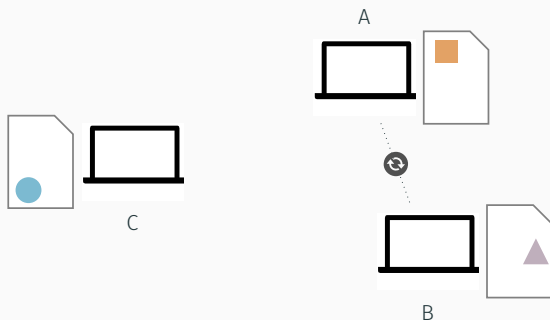
[1]. TERRY et al., « Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System ».

Réplication dans applications collaboratives pair-à-pair



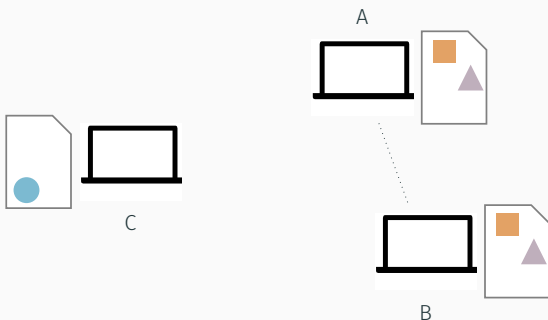
[1]. TERRY et al., « Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System ».

Réplication dans applications collaboratives pair-à-pair



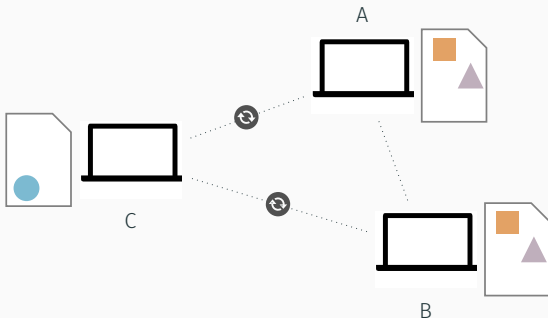
[1]. TERRY et al., « Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System ».

Réplication dans applications collaboratives pair-à-pair



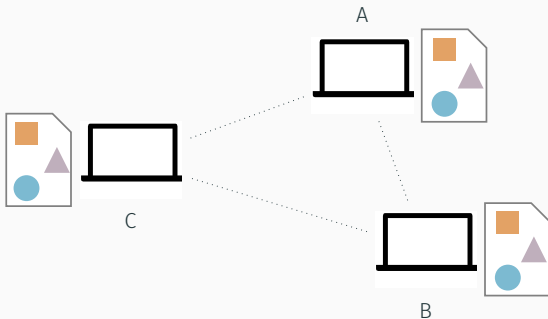
[1]. TERRY et al., « Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System ».

Réplication dans applications collaboratives pair-à-pair



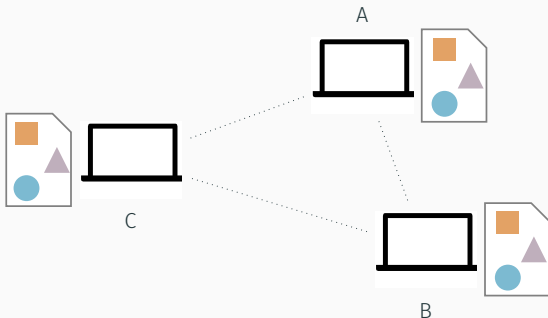
[1]. TERRY et al., « Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System ».

Réplication dans applications collaboratives pair-à-pair



[1]. TERRY et al., « Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System ».

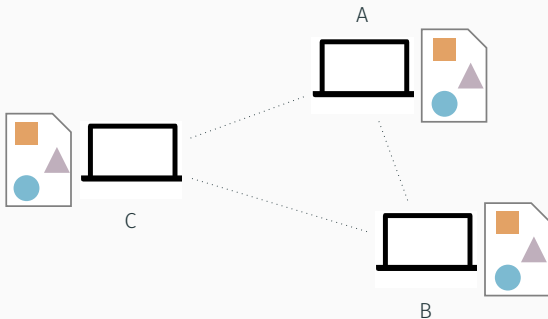
Réplication dans applications collaboratives pair-à-pair



- Doit garantir **convergence à terme**^[1]

[1]. TERRY et al., « Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System ».

Réplication dans applications collaboratives pair-à-pair



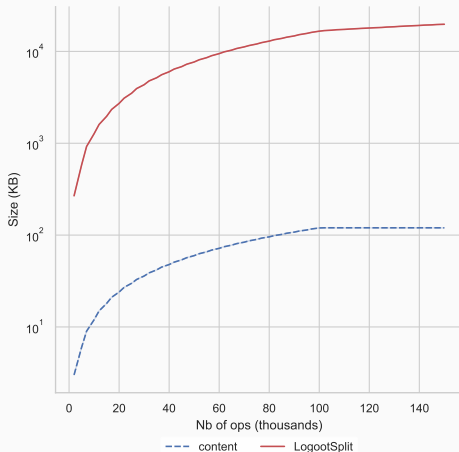
- Doit garantir **convergence à terme**^[1]

Nécessite mécanismes de résolution de conflits

[1]. TERRY et al., « Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System ».

Évaluation de MUTE

Taille du texte comparée à taille de la séquence répliquée



- 1% contenu...
- ...99% métadonnées

Comment peut-on réduire le surcoût mémoire
des mécanismes de résolution de conflits dans
les applications pair-à-pair ?

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

- Nouvelles spécifications des types de données, e.g. *Ensemble* ou *Séquence*
- Incorpore nativement mécanisme de résolution de conflits

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

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

- Nouvelles spécifications des types de données, e.g. *Ensemble* ou *Séquence*
- Incorpore nativement mécanisme de résolution de conflits

Propriétés des CRDTs

- Permettent modifications **sans coordination**
- Garantissent la **convergence forte**

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

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

- Nouvelles spécifications des types de données, e.g. *Ensemble* ou *Séquence*
- Incorpore nativement mécanisme de résolution de conflits

Propriétés des CRDTs

- 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 ».

LogootSplit^[4], un CRDT pour le type Séquence

- Assigne **identifiant de position** à chaque élément de la séquence

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

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

LogootSplit^[4], un CRDT pour le type Séquence

- Assigne **identifiant de position** à chaque élément de la séquence

Propriétés des identifiants de position^[3]

1. Unique
2. Immuable
3. Ordonnable par une relation d'ordre strict total $<_{id}$
4. Appartenant à un espace dense

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

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

LogootSplit^[4], un CRDT pour le type Séquence

- Assigne **identifiant de position** à chaque élément de la séquence

Propriétés des identifiants de position^[3]

1. Unique
2. Immuable
3. Ordonnable par une relation d'ordre strict total $<_{id}$
4. Appartenant à un espace dense

- **Ordonne les éléments** entre eux **en utilisant** leurs **identifiants**

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

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

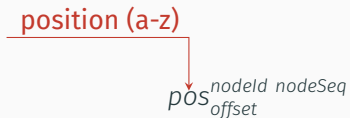
Identifiant

- Composé d'un ou plusieurs tuples suivants

$pos_{offset}^{nodeId \ nodeSeq}$

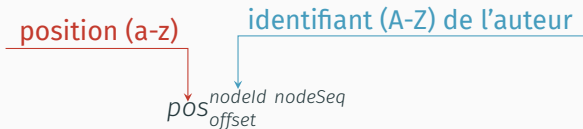
Identifiant

- Composé d'un ou plusieurs tuples suivants



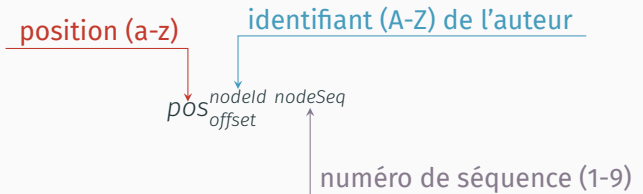
Identifiant

- Composé d'un ou plusieurs tuples suivants



Identifiant

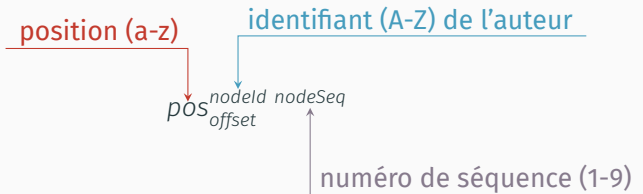
- Composé d'un ou plusieurs tuples suivants



Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples suivants



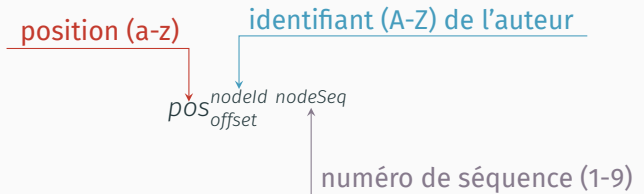
Exemples

d_0^{F5}

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples suivants



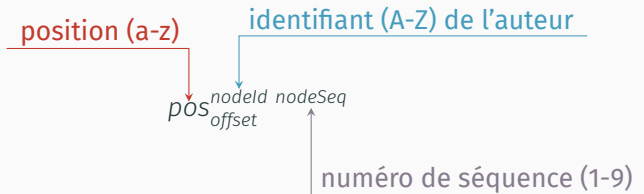
Exemples

$$d_0^{F5} <_{id} m_0^{C1}$$

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples suivants



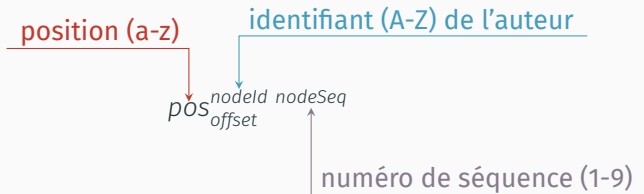
Exemples

$$d_0^{F5} <_{id} m_0^{C1} <_{id} m_0^{C1} f_0^{E1}$$

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples suivants



Exemples

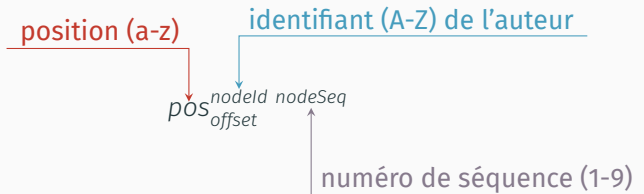
$$d_0^{F5} <_{id} m_0^{C1} <_{id} m_0^{C1} f_0^{E1}$$

$$i_0^{B1} <_{id} ? <_{id} i_1^{B1}$$

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples suivants



Exemples

$$d_0^{F5} <_{id} m_0^{C1} <_{id} m_0^{C1} f_0^{E1}$$

$$i_0^{B1} <_{id} i_0^{B1} f_0^{A1} <_{id} i_1^{B1}$$

- Coûteux de stocker les identifiants de chaque élément

B	A	N	J	O
m_0^{C1}	m_1^{C1}	m_2^{C1}	m_3^{C1}	m_4^{C1}

Bloc LogootSplit

- Coûteux de stocker les identifiants de chaque élément

B	A	N	J	O
m_0^{C1}	m_1^{C1}	m_2^{C1}	m_3^{C1}	m_4^{C1}

- Aggrège en un **bloc** éléments ayant **identifiants contigus**

Identifiants contigus

Deux identifiants sont contigus si et seulement si les deux identifiants sont identiques à l'exception de leur dernier offset et que leur derniers offsets sont consécutifs.

Bloc LogootSplit

- Coûteux de stocker les identifiants de chaque élément

B	A	N	J	O
m_0^{C1}	m_1^{C1}	m_2^{C1}	m_3^{C1}	m_4^{C1}

- Aggrège en un **bloc** éléments ayant **identifiants contigus**

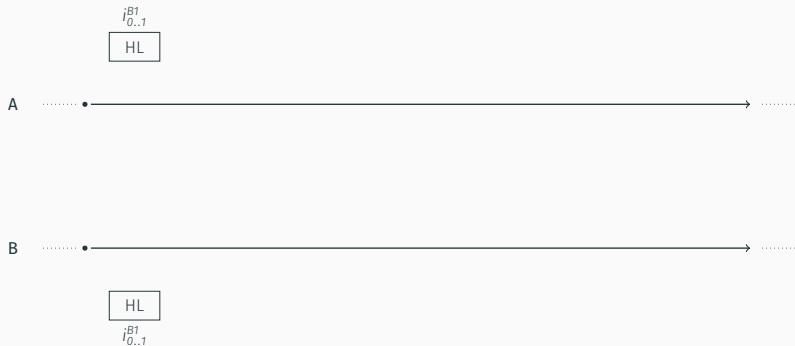
Identifiants contigus

Deux identifiants sont contigus si et seulement si les deux identifiants sont identiques à l'exception de leur dernier offset et que leur derniers offsets sont consécutifs.

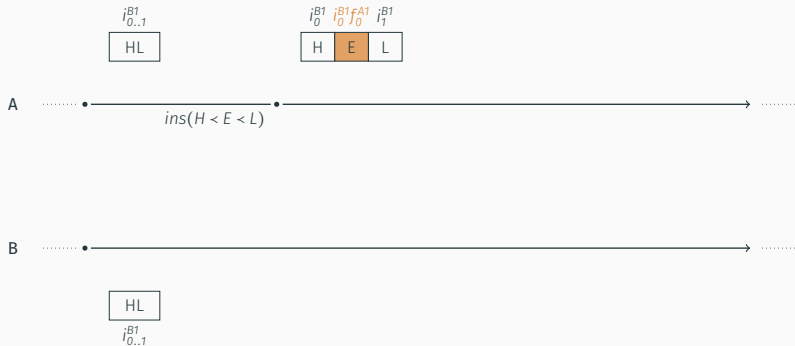
- Note l'intervalle d'identifiants d'un bloc : $pos_{begin..end}^{nodeId nodeSeq}$

BANJO
$m_{0..4}^{C1}$

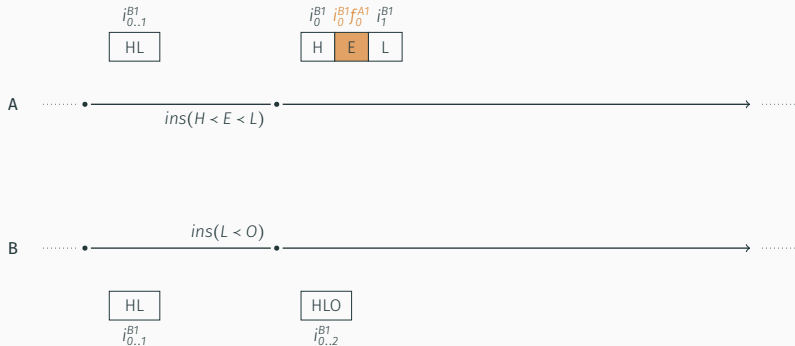
Exemple insertions concurrentes



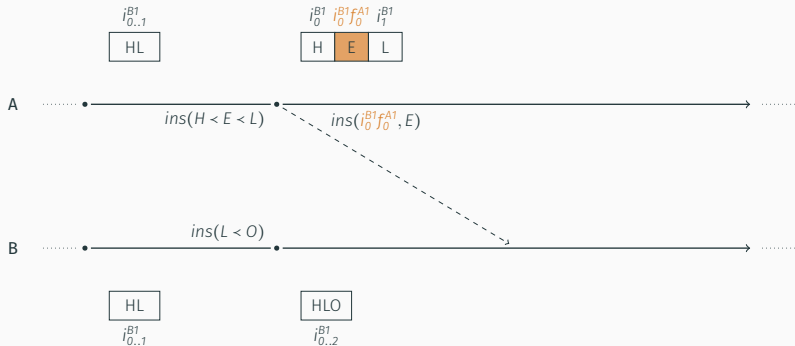
Exemple insertions concurrentes



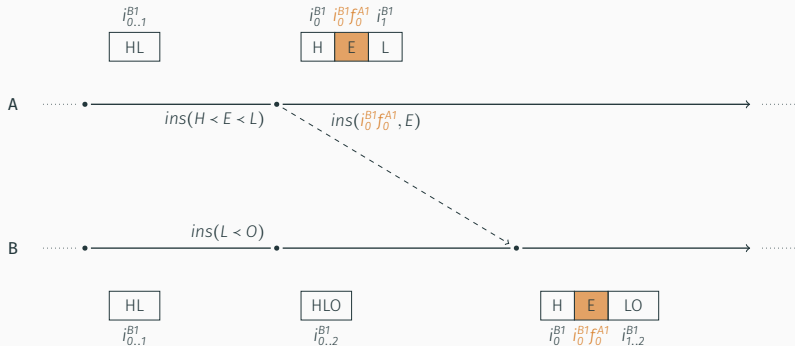
Exemple insertions concurrentes



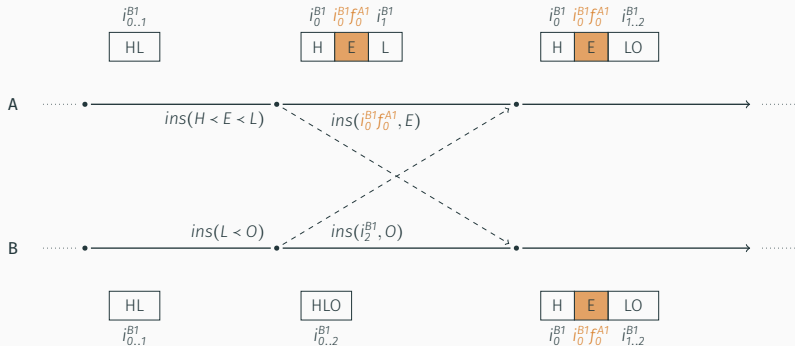
Exemple insertions concurrentes



Exemple insertions concurrentes



Exemple insertions concurrentes



Sources croissance métadonnées

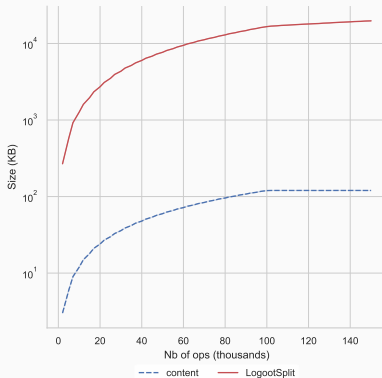
- Croissance non-bornée de la taille des identifiants
- Fragmentation en blocs courts

Limites de LogootSplit

Sources croissance métadonnées

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

Taille du contenu comparé à la taille de la séquence LogootSplit



L'approche core-nebula^[5]

- Ré-assigne des identifiants courts aux éléments, c.-à-d. les *renomme*
- Transforme les opérations *insert* et *remove* concurrentes...

[5]. ZAWIRSKI et al., « Asynchronous rebalancing of a replicated tree ».

L'approche core-nebula^[5]

- Ré-assigne des identifiants courts aux éléments, c.-à-d. les *renomme*
- Transforme les opérations *insert* et *remove* concurrentes...
- ...mais ne supportent pas opérations *rename* concurrentes

[5]. ZAWIRSKI et al., « Asynchronous rebalancing of a replicated tree ».

L'approche core-nebula^[5]

- Ré-assigne des identifiants courts aux éléments, c.-à-d. les *renomme*
- Transforme les opérations *insert* et *remove* concurrentes...
- ...mais ne supportent pas opérations *rename* concurrentes

Inadaptée aux applications pair-à-pair

[5]. ZAWIRSKI et al., « Asynchronous rebalancing of a replicated tree ».

Proposition

Mécanisme de renommage supportant les
renommages concurrents

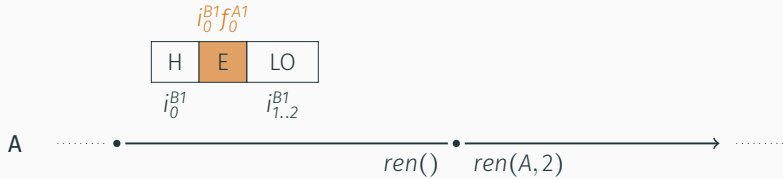
RenamableLogootSplit

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

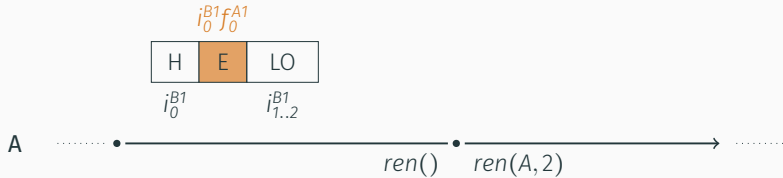
Propriétés de l'opération *rename*

- 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*

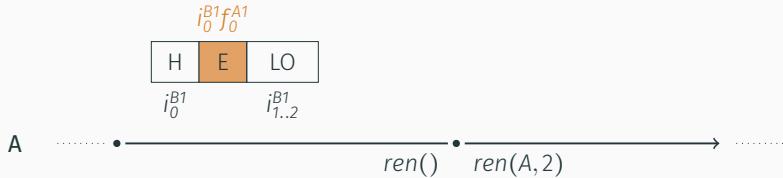


Opération *rename*



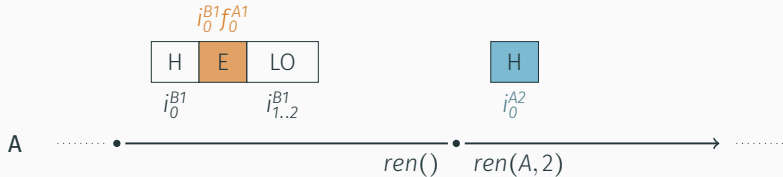
- Génère nouvel identifiant pour le 1er élément :

Opération *rename*



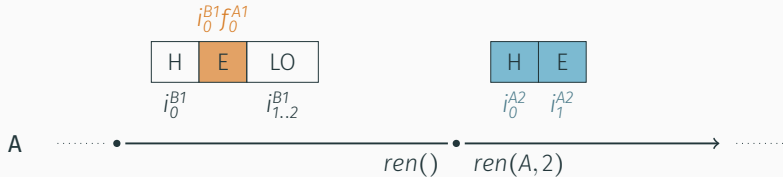
- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$

Opération *rename*



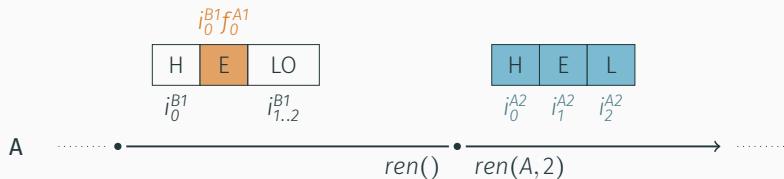
- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants :

Opération *rename*



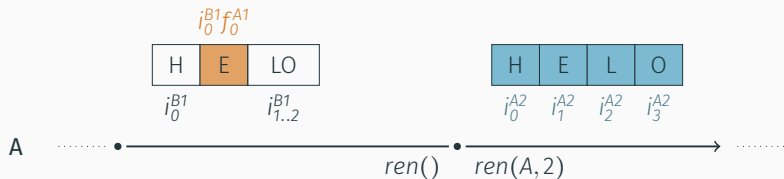
- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants : i_1^{A2}

Opération *rename*



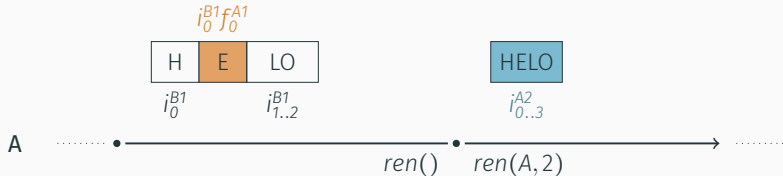
- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants : i_1^{A2}, i_2^{A2}

Opération *rename*



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

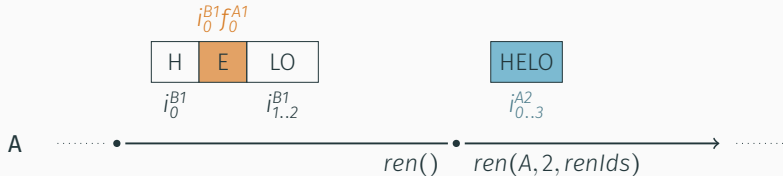
Opération *rename*



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

Regroupe tous les éléments en 1 unique bloc

Opération *rename*

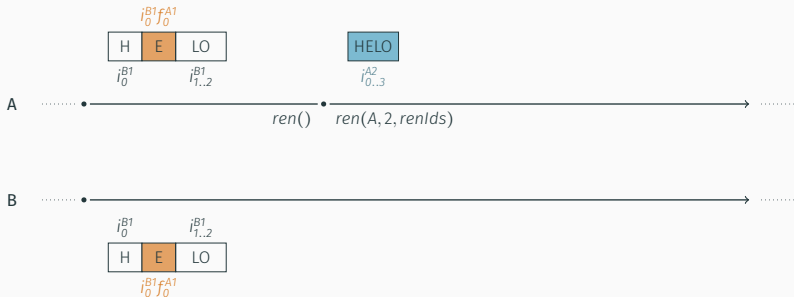


- Génère nouvel identifiant pour le 1er élément : $i_0^{B1} \rightarrow i_0^{A2}$
- Puis génère identifiants contigus pour éléments suivants : $i_1^{A2}, i_2^{A2}, \dots$

Regroupe tous les éléments en 1 unique bloc

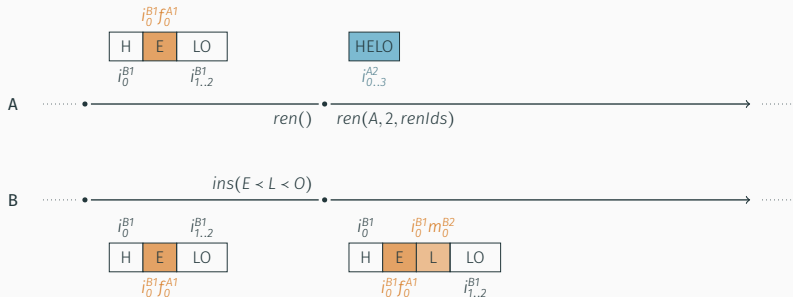
- Stocke identifiants ($[i_0^{B1}, i_0^{B1} f_0^{A1}, \dots]$) de l'état d'origine : *renlds*

Interactions avec opérations *insert* et *remove* concurrentes



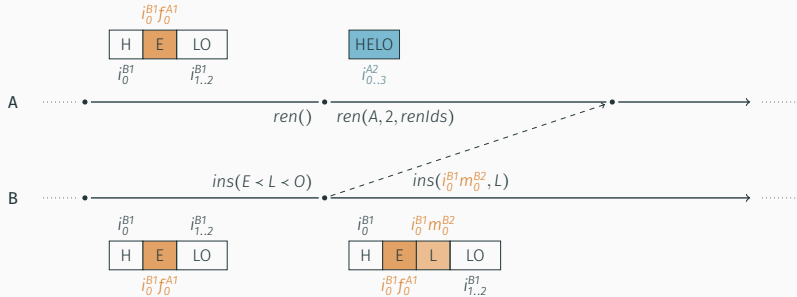
- Peuvent générer opérations concurrentes aux opérations *rename*

Interactions avec opérations *insert* et *remove* concurrentes



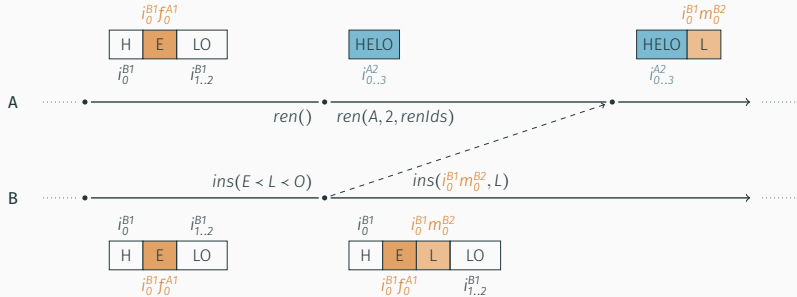
- Peuvent générer opérations concurrentes aux opérations *rename*

Interactions avec opérations *insert* et *remove* concurrentes



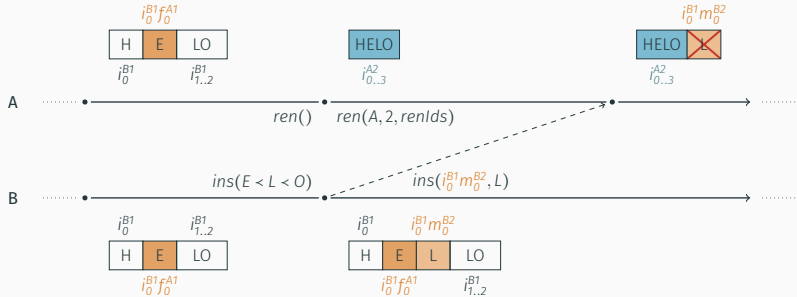
- Peuvent générer opérations concurrentes aux opérations *rename*

Interactions avec opérations *insert* et *remove* concurrentes



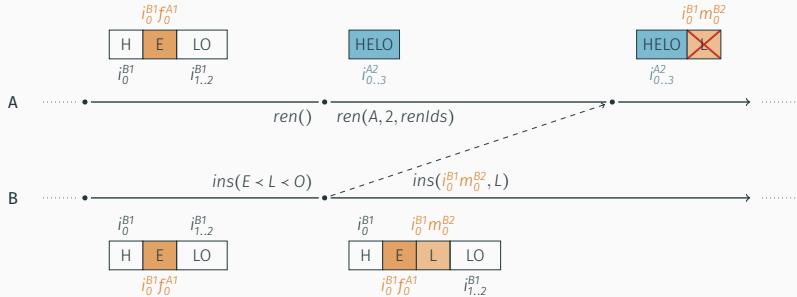
- Peuvent générer opérations concurrentes aux opérations *rename*

Interactions avec opérations *insert* et *remove* concurrentes



- Peuvent générer opérations concurrentes aux opérations *rename*
- Produisent anomalies si intégrées naïvement

Interactions avec opérations *insert* et *remove* concurrentes



- Peuvent générer opérations concurrentes aux opérations *rename*
- Produisent anomalies si intégrées naïvement

Nécessité d'un mécanisme dédié

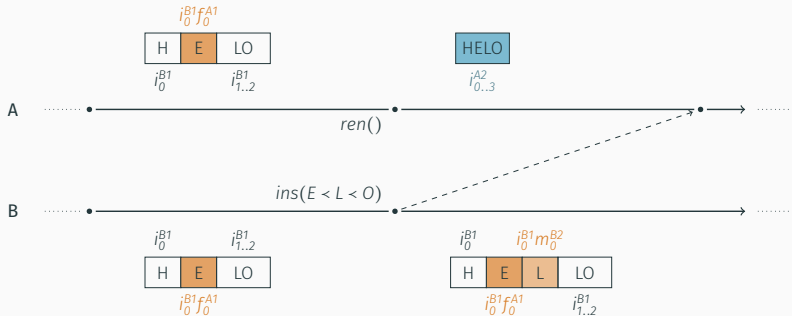
Besoins

1. Détecter les opérations concurrentes aux opérations *rename*
2. Prendre en compte effet des opérations *rename* lors de l'intégration des opérations concurrentes

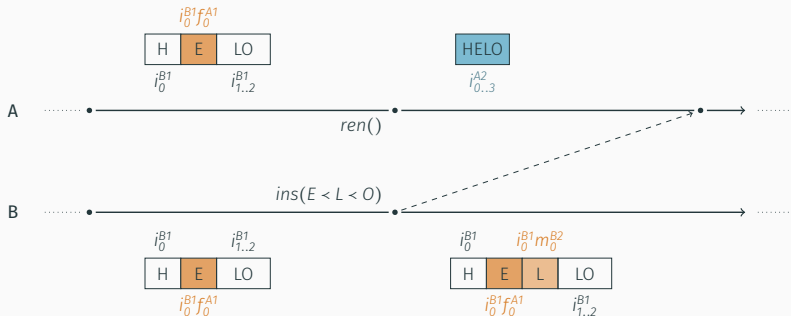
Besoins

1. Détecter les opérations concurrentes aux opérations *rename*
2. Prendre en compte effet des opérations *rename* lors de l'intégration des opérations concurrentes

Détection des opérations concurrentes à opération *rename*

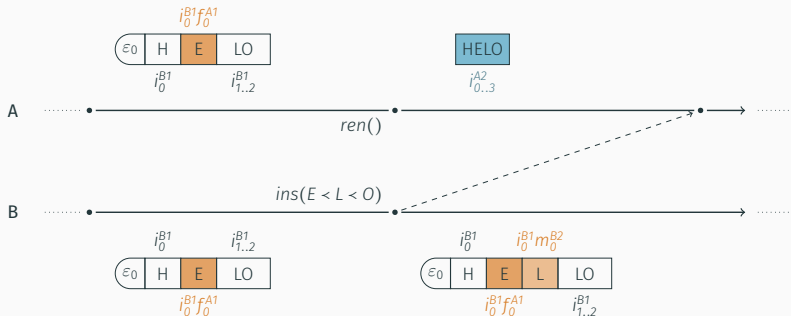


Détection des opérations concurrentes à opération *rename*



Ajout mécanisme d'époques

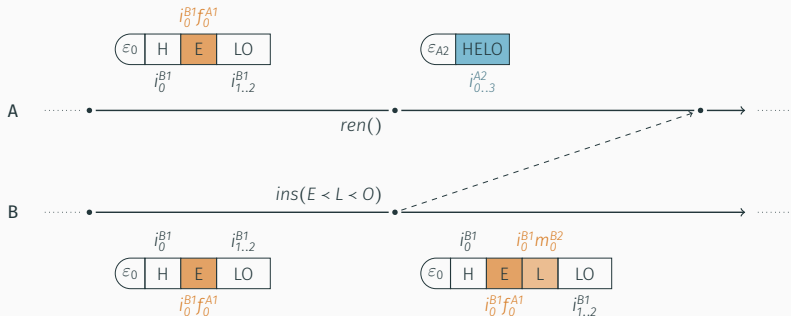
Détection des opérations concurrentes à opération *rename*



Ajout mécanisme d'époques

- Séquence commence à époque d'origine, notée ϵ_0

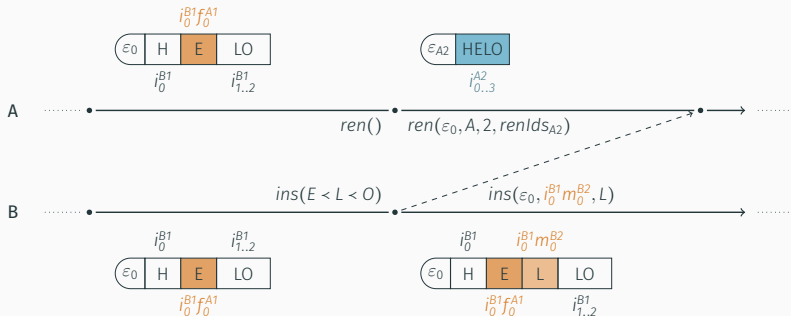
Détection des opérations concurrentes à opération *rename*



Ajout mécanisme d'époques

- Séquence commence à époque d'origine, notée ϵ_0
- *rename* font progresser à nouvelle époque, $\epsilon_{nodeId \ nodeSeq}$

Détection des opérations concurrentes à opération *rename*



Ajout mécanisme d'époques

- Séquence commence à époque d'origine, notée ϵ_0
- *rename* font progresser à nouvelle époque, $\epsilon_{nodeId \ nodeSeq}$
- Opérations labellisées avec époque de génération

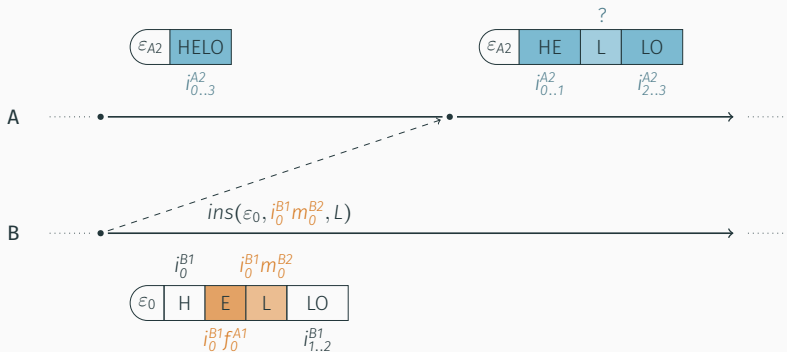
Besoins

1. Détecter les opérations concurrentes aux opérations *rename*
2. Prendre en compte effet des opérations *rename* lors de l'intégration des opérations concurrentes

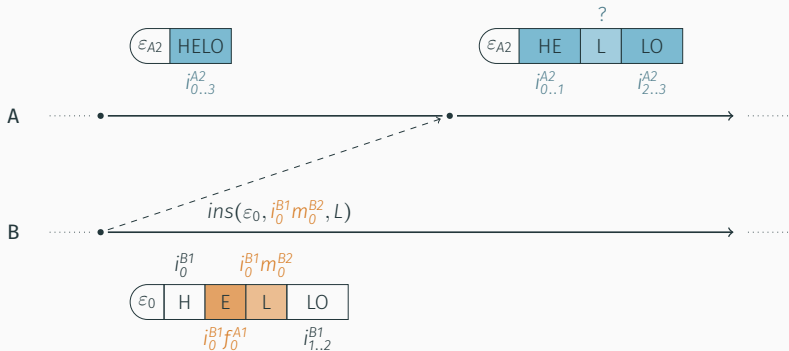
Besoins

1. Détecter les opérations concurrentes aux opérations *rename*
2. Prendre en compte effet des opérations *rename* lors de l'intégration des opérations concurrentes

Intégration des opérations *insert* et *remove* concurrentes

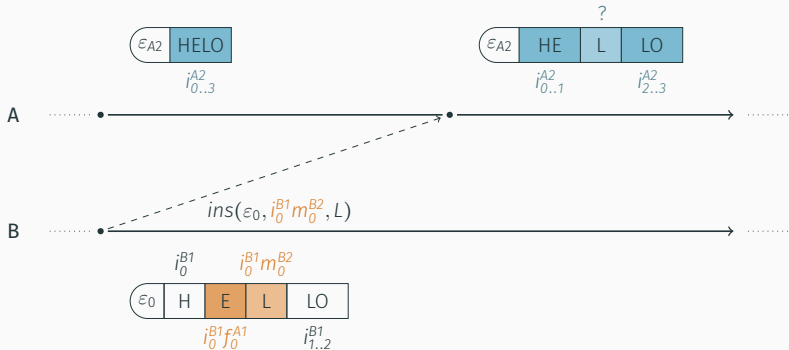


Intégration des opérations *insert* et *remove* concurrentes



Ajout d'un mécanisme de transformation des opérations *insert* et *remove* concurrentes

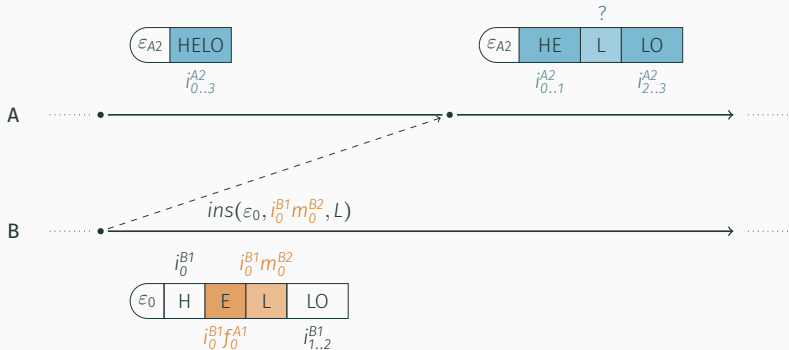
Intégration des opérations *insert* et *remove* concurrentes



Ajout d'un mécanisme de transformation des opérations *insert* et *remove* concurrentes

- Prend la forme de l'algorithme `renameId`

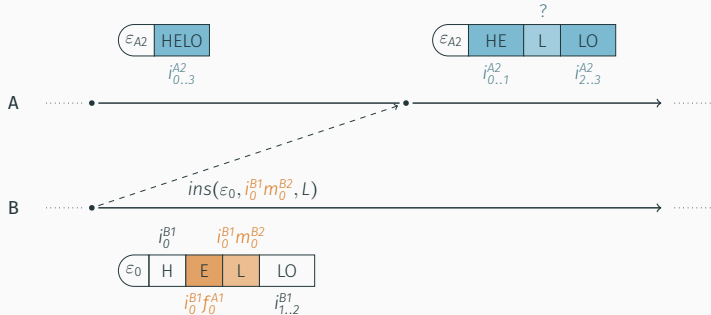
Intégration des opérations *insert* et *remove* concurrentes



Ajout d'un mécanisme de transformation des opérations *insert* et *remove* concurrentes

- Prend la forme de l'algorithme `renameId`
- Inclure l'effet de l'opération *rename* dans l'opération transformée

Exemple de renameId

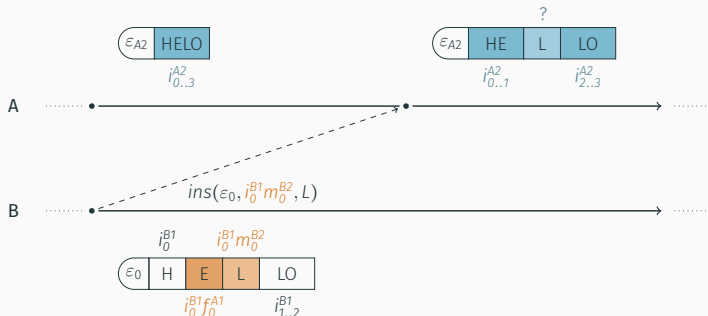


Rappel :

$$renIds_{A2} = [i_0^{B1}, i_0^{B1} f_0^{A1}, i_1^{B1}, i_2^{B1}]$$

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

Exemple de renameId



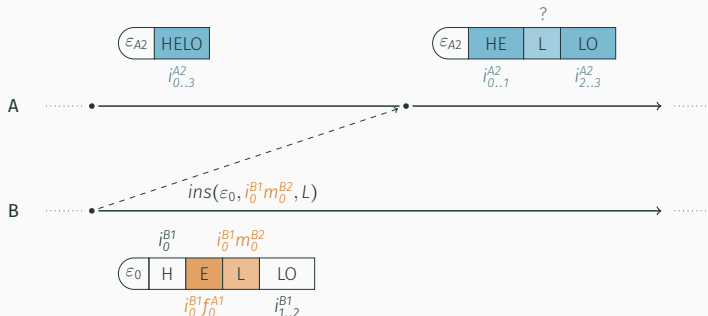
Rappel :

$$renIds_{A2} = [i_0^{B1}, i_0^{B1} f_0^{A1}, i_1^{B1}, i_2^{B1}]$$

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

- Trouver son prédécesseur dans $renIds_{A2}$: $i_0^{B1} f_0^{A1}$

Exemple de renameId



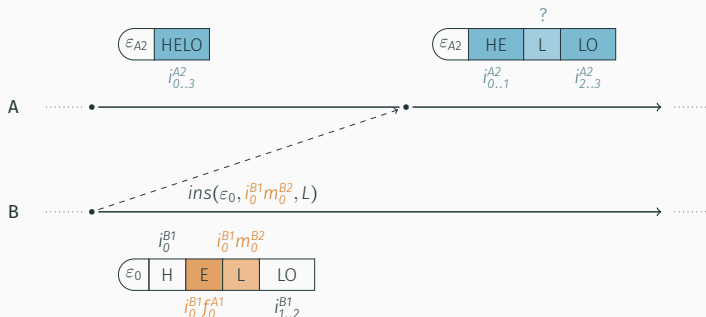
Rappel :

$$renIds_{A2} = [i_0^{B1}, i_0^{B1} f_0^{A1}, i_1^{B1}, i_2^{B1}]$$

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

- Trouver son prédécesseur dans $renIds_{A2}$: $i_0^{B1} f_0^{A1}$
- Utiliser son index (1) pour trouver équivalent à époque ϵ_{A2} : i_1^{A2}

Exemple de renameId



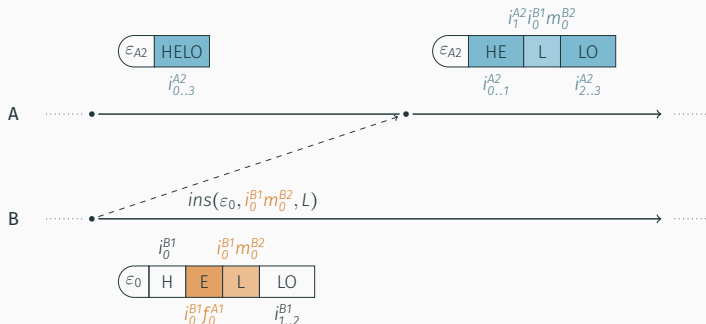
Rappel :

$$renIds_{A2} = [i_0^{B1}, i_0^{B1} f_0^{A1}, i_1^{B1}, i_2^{B1}]$$

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

- Trouver son prédécesseur dans $renIds_{A2}$: $i_0^{B1} f_0^{A1}$
- Utiliser son index (1) pour trouver équivalent à époque ϵ_{A2} : i_1^{A2}
- Préfixer $i_0^{B1} m_0^{B2}$ par ce dernier : $i_1^{A2} i_0^{B1} m_0^{B2}$

Exemple de renameId



Rappel :

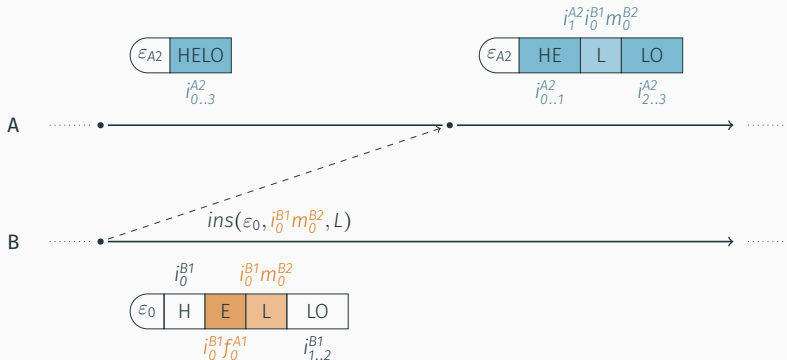
$$renIds_{A2} = [i_0^{B1}, i_0^{B1} f_0^{A1}, i_1^{B1}, i_2^{B1}]$$

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

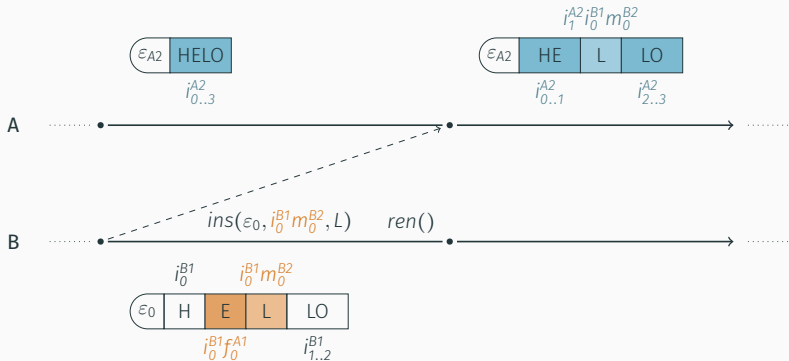
- Trouver son prédécesseur dans $renIds_{A2}$: $i_0^{B1} f_0^{A1}$
- Utiliser son index (1) pour trouver équivalent à époque ϵ_{A2} : i_1^{A2}
- Préfixer $i_0^{B1} m_0^{B2}$ par ce dernier : $i_1^{A2} i_0^{B1} m_0^{B2}$

Et en cas d'opérations *rename* concurrentes?

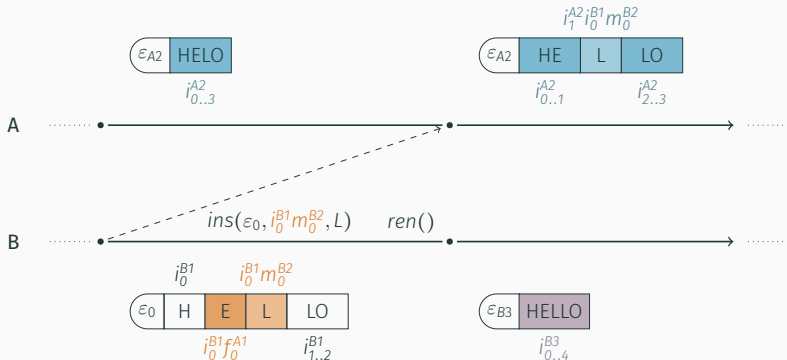
Opérations *rename* concurrentes



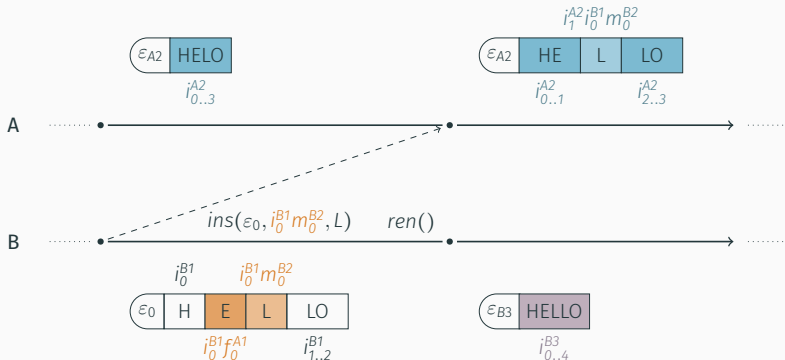
Opérations *rename* concurrentes



Opérations *rename* concurrentes

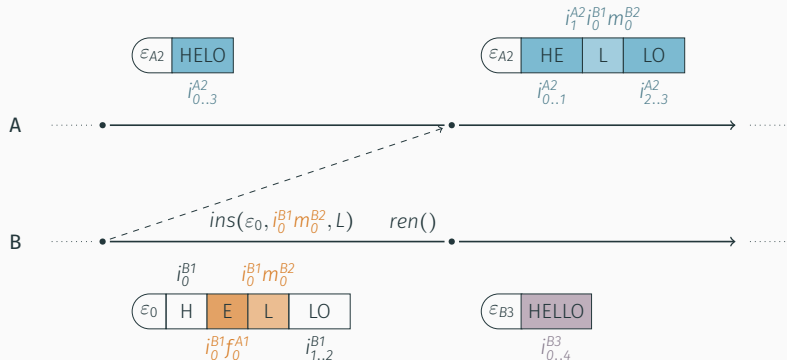


Opérations *rename* concurrentes



Comment faire converger les noeuds ?

Opérations *rename* concurrentes



Comment faire converger les noeuds ?

Besoin d'un mécanisme additionnel de résolution de conflits

Observation

- Opérations *rename* sont des opérations systèmes...
- ...pas des opérations utilisateur-rices

Résolution de conflits entre opérations *rename* concurrentes

Observation

- Opérations *rename* sont des opérations systèmes...
- ...pas des opérations utilisateur-rices

Proposition

- Considérer une opération *rename* comme prioritaire...
- ...et ignorer les opérations *rename* en conflit avec elle

Algorithme d'intégration d'une opération *rename*

Intuition

Algorithme d'intégration d'une opération *rename*

Intuition

1. Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues

Algorithme d'intégration d'une opération *rename*

Intuition

1. Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
2. Choisir entre époque courante et nouvelle époque **l'époque cible**

Algorithme d'intégration d'une opération *rename*

Intuition

1. Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
2. Choisir entre époque courante et nouvelle époque **l'époque cible**
3. Si changement d'époque cible

Algorithme d'intégration d'une opération *rename*

Intuition

1. Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
2. Choisir entre époque courante et nouvelle époque **l'époque cible**
3. Si changement d'époque cible
 - 3.1 Calculer chemin entre époque courante et époque cible, et notamment leur Plus Proche Ancêtre Commun (PPAC)

Algorithme d'intégration d'une opération *rename*

Intuition

1. Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
2. Choisir entre époque courante et nouvelle époque **l'époque cible**
3. Si changement d'époque cible
 - 3.1 Calculer chemin entre époque courante et époque cible, et notamment leur Plus Proche Ancêtre Commun (PPAC)
 - 3.2 Annuler l'effet des opérations *rename* de l'époque courante au PPAC

Algorithme d'intégration d'une opération *rename*

Intuition

1. Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
2. Choisir entre époque courante et nouvelle époque **l'époque cible**
3. Si changement d'époque cible
 - 3.1 Calculer chemin entre époque courante et époque cible, et notamment leur Plus Proche Ancêtre Commun (PPAC)
 - 3.2 Annuler l'effet des opérations *rename* de l'époque courante au PPAC
 - 3.3 Appliquer l'effet des opérations *rename* du PPAC à l'époque cible

Algorithme d'intégration d'une opération *rename*

Intuition

1. Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
2. Choisir entre époque courante et nouvelle époque **l'époque cible**
3. Si changement d'époque cible
 - 3.1 Calculer chemin entre époque courante et époque cible, et notamment leur Plus Proche Ancêtre Commun (PPAC)
 - 3.2 Annuler l'effet des opérations *rename* de l'époque courante au PPAC
 - 3.3 Appliquer l'effet des opérations *rename* du PPAC à l'époque cible

Algorithme d'intégration d'une opération *rename*

Intuition

1. Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
2. Choisir entre époque courante et nouvelle époque l'époque cible
3. Si changement d'époque cible
 - 3.1 Calculer chemin entre époque courante et époque cible, et notamment leur Plus Proche Ancêtre Commun (PPAC)
 - 3.2 Annuler l'effet des opérations *rename* de l'époque courante au PPAC
 - 3.3 Appliquer l'effet des opérations *rename* du PPAC à l'époque cible

Choisir une époque comme époque cible

A →

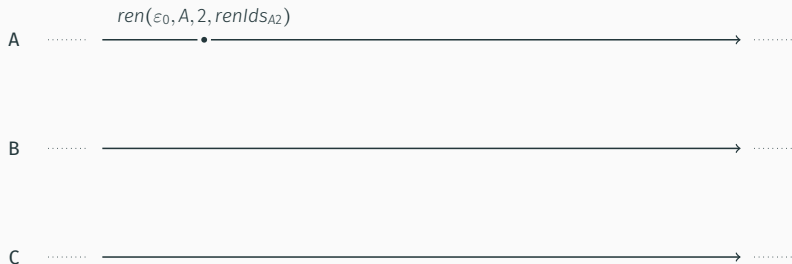
B →

C →

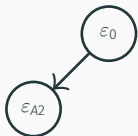
Arbre des époques



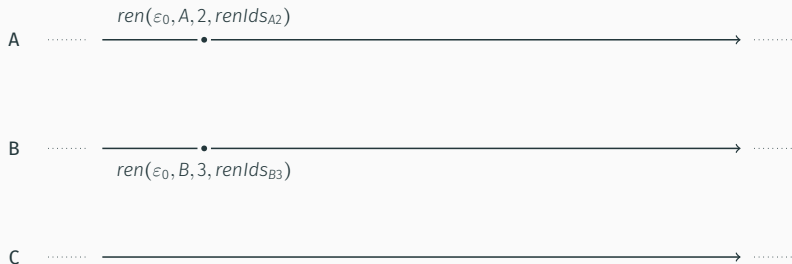
Choisir une époque comme époque cible



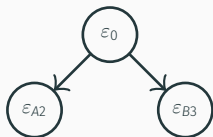
Arbre des époques



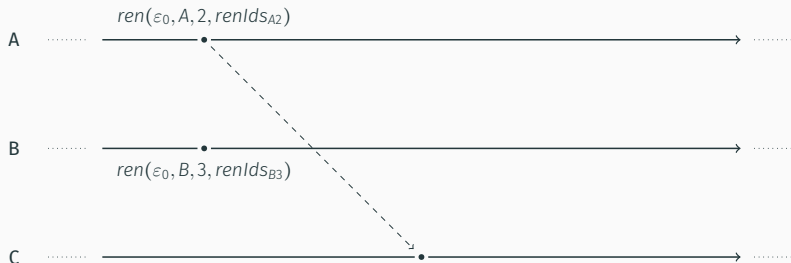
Choisir une époque comme époque cible



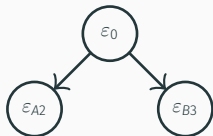
Arbre des époques



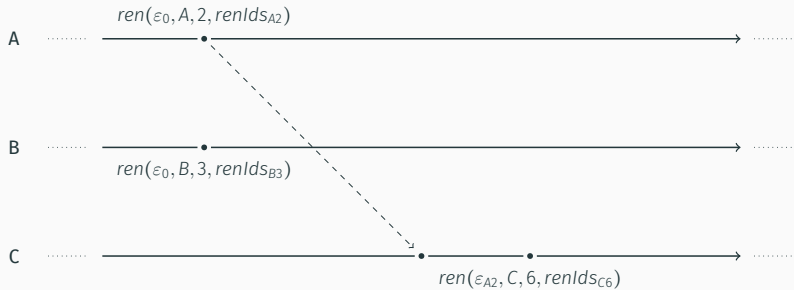
Choisir une époque comme époque cible



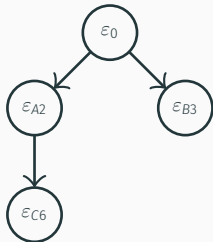
Arbre des époques



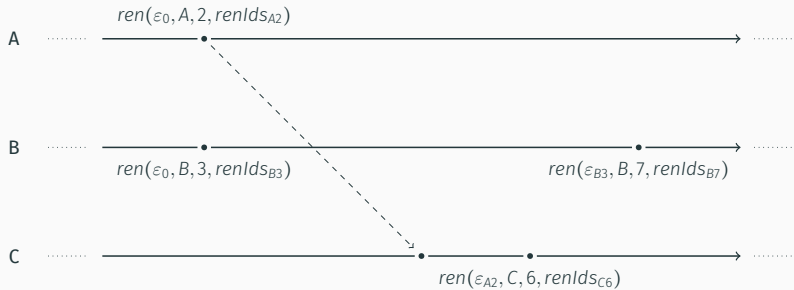
Choisir une époque comme époque cible



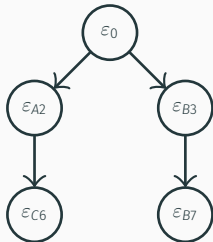
Arbre des époques



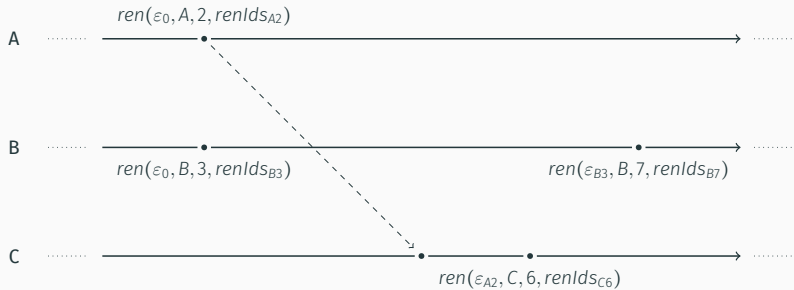
Choisir une époque comme époque cible



Arbre des époques

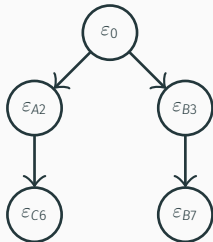


Choisir une époque comme époque cible

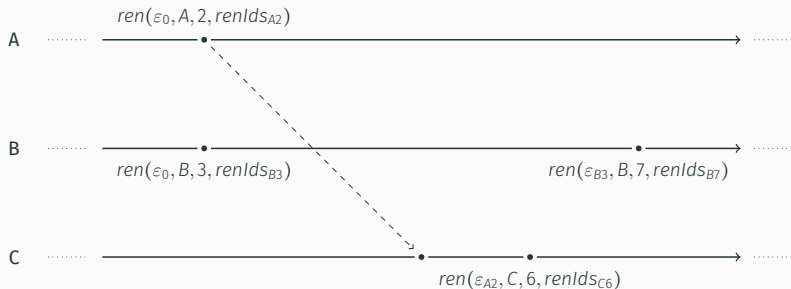


Arbre des époques

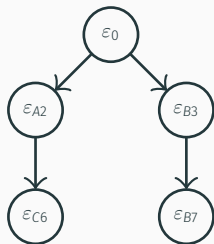
Comment choisir?



Choisir une époque comme époque cible



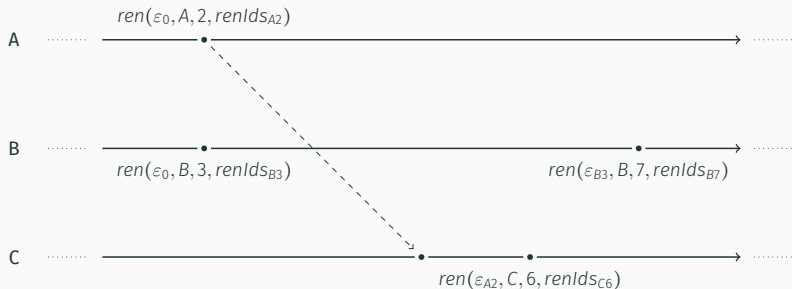
Arbre des époques



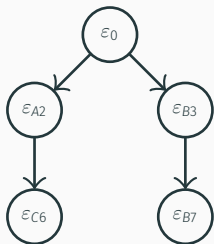
Comment choisir?

- Définit relation *priority*, notée $<_{\varepsilon}$, ordre strict total sur les époques

Choisir une époque comme époque cible



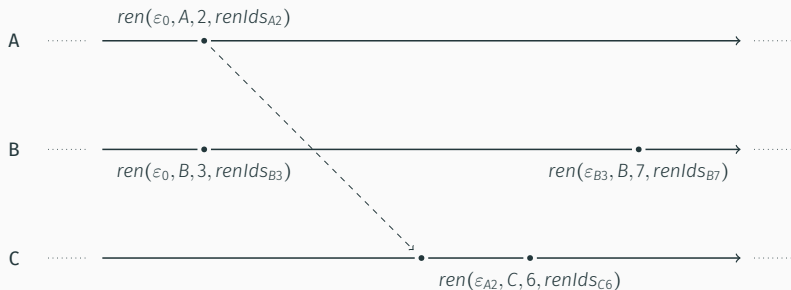
Arbre des époques



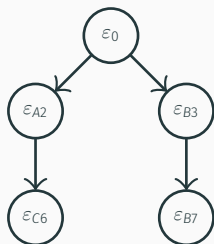
Comment choisir?

- Définit relation *priority*, notée $<_{\epsilon}$, ordre strict total sur les époques
- Utilise **ordre lexicographique sur chemins** des époques dans l'arbre

Choisir une époque comme époque cible



Arbre des époques



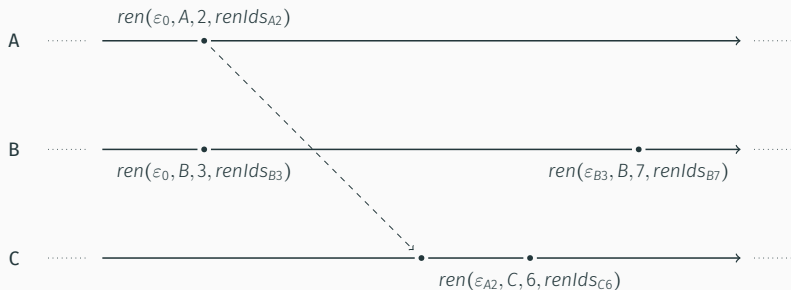
Comment choisir?

- Définit relation *priority*, notée $<_{\epsilon}$, ordre strict total sur les époques
- Utilise **ordre lexicographique sur chemins** des époques dans l'arbre

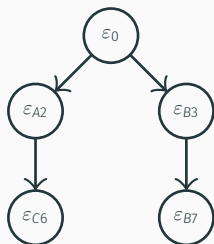
Exemple

$$\epsilon_0 < \epsilon_0 \epsilon_{A2}$$

Choisir une époque comme époque cible



Arbre des époques



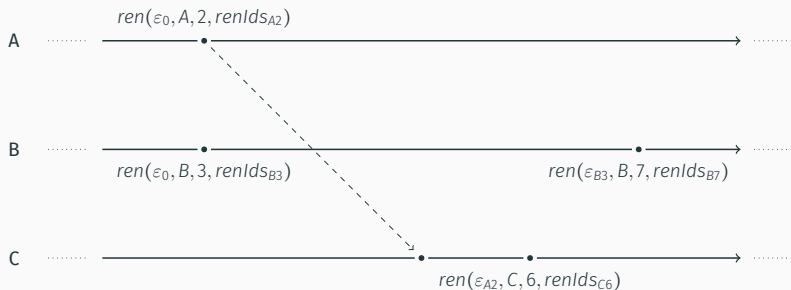
Comment choisir?

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

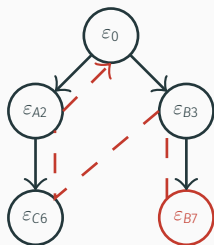
Exemple

$$\varepsilon_0 < \varepsilon_0 \varepsilon_{A2} < \varepsilon_0 \varepsilon_{A2} \varepsilon_{C6}$$

Choisir une époque comme époque cible



Arbre des époques



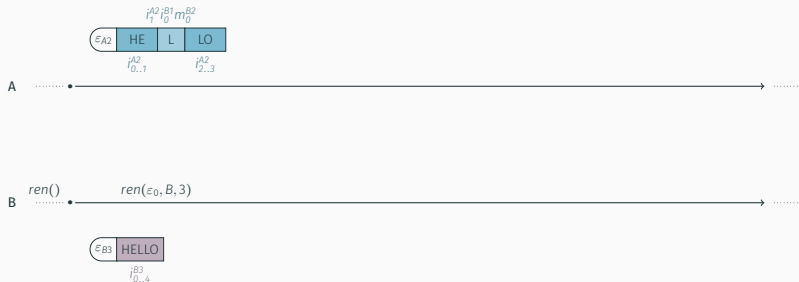
Comment choisir?

- Définit relation *priority*, notée $<_{\epsilon}$, ordre strict total sur les époques
- Utilise **ordre lexicographique sur chemins** des époques dans l'arbre

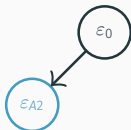
Exemple

$$\epsilon_0 < \epsilon_0 \epsilon_{A2} < \epsilon_0 \epsilon_{A2} \epsilon_{C6} < \epsilon_0 \epsilon_{B3} \epsilon_{B7}$$

Exemple - Calculs des transformations à effectuer



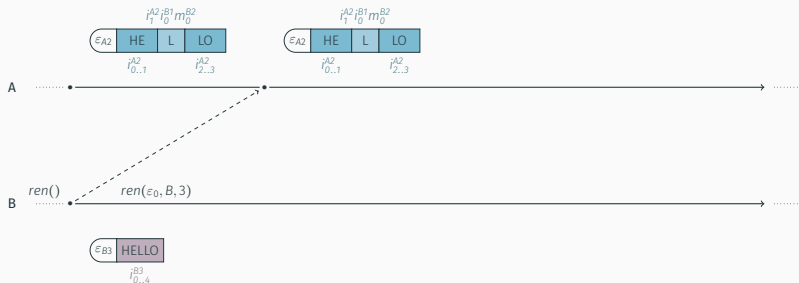
Arbre des époques de A



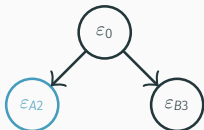
Étapes

- Époque courante : ϵ_{A2}

Exemple - Calculs des transformations à effectuer



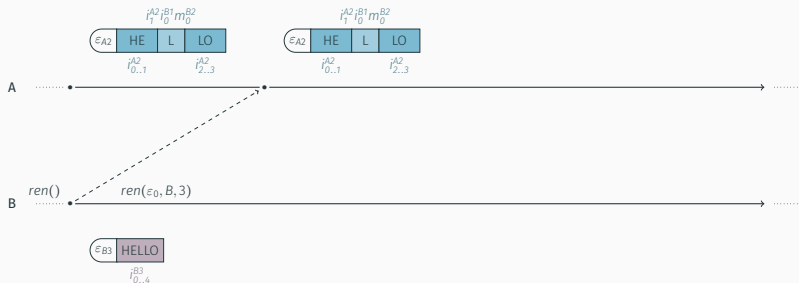
Arbre des époques de A



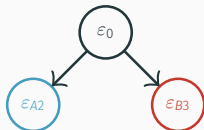
Étapes

- Époque courante : ϵ_{A2}

Exemple - Calculs des transformations à effectuer



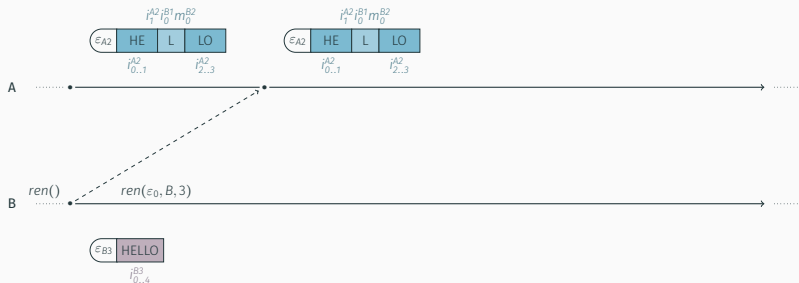
Arbre des époques de A



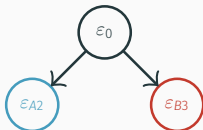
Étapes

- Époque courante : ϵ_{A2}
- Époque cible : ϵ_{B3}

Exemple - Calculs des transformations à effectuer



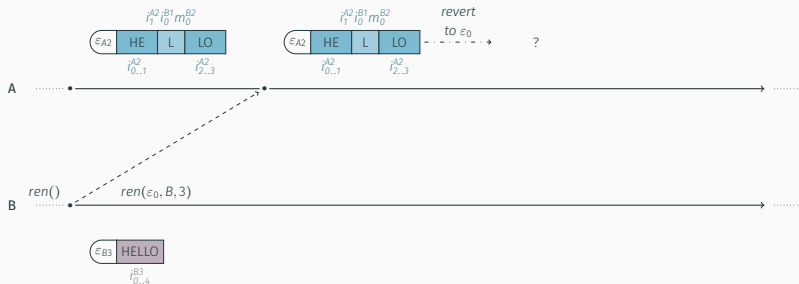
Arbre des époques de A



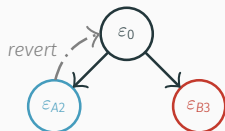
Étapes

- Époque courante : ε_{A2}
- Époque cible : ε_{B3}
- Plus Proche Ancêtre Commun : ε_0

Exemple - Calculs des transformations à effectuer



Arbre des époques de A

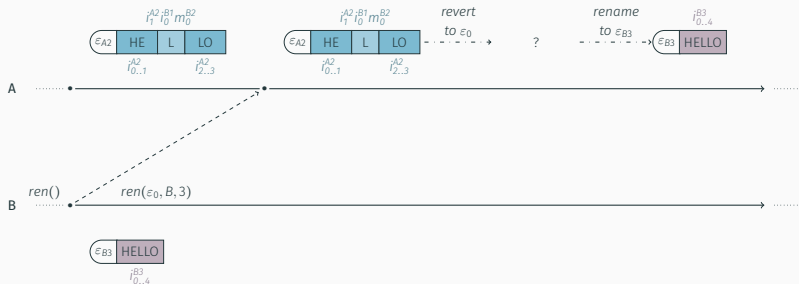


Étapes

- Époque courante : ϵ_{A2}
- Époque cible : ϵ_{B3}
- Plus Proche Ancêtre Commun : ϵ_0

Doit annuler ϵ_{A2}

Exemple - Calculs des transformations à effectuer



Arbre des époques de A



Étapes

- Époque courante : ϵ_{A2}
- Époque cible : ϵ_{B3}
- Plus Proche Ancêtre Commun : ϵ_0

Doit annuler ϵ_{A2} puis appliquer ϵ_{B3}

Algorithme d'intégration d'une opération *rename*

Intuition

1. Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
2. Choisir entre époque courante et nouvelle époque l'époque cible
3. Si changement d'époque cible
 - 3.1 Calculer chemin entre époque courante et époque cible, et notamment leur Plus Proche Ancêtre Commun (PPAC)
 - 3.2 Annuler l'effet des opérations *rename* de l'époque courante au PPAC
 - 3.3 Appliquer l'effet des opérations *rename* du PPAC à l'époque cible

Algorithme d'intégration d'une opération *rename*

Intuition

1. Ajouter l'époque créée par l'opération *rename* à l'ensemble des époques connues
2. Choisir entre époque courante et nouvelle époque l'époque cible
3. Si changement d'époque cible
 - 3.1 Calculer chemin entre époque courante et époque cible, et notamment leur Plus Proche Ancêtre Commun (PPAC)
 - 3.2 Annuler l'effet des opérations *rename* de l'époque courante au PPAC
 - 3.3 Appliquer l'effet des opérations *rename* du PPAC à l'époque cible

Annuler l'effet d'une opération *rename*

Ajout d'un nouveau mécanisme de transformation

Annuler l'effet d'une opération *rename*

Ajout d'un nouveau mécanisme de transformation

- Prend la forme de l'algorithme *revertRenameId*

Annuler l'effet d'une opération *rename*

Ajout d'un nouveau mécanisme de transformation

- Prend la forme de l'algorithme *revertRenameId*
- Exclure l'effet de l'opération *rename*

Annuler l'effet d'une opération *rename*

Ajout d'un nouveau mécanisme de transformation

- Prend la forme de l'algorithme *revertRenamed*
- Exclure l'effet de l'opération *rename*

Intuition

Annuler l'effet d'une opération *rename*

Ajout d'un nouveau mécanisme de transformation

- Prend la forme de l'algorithme *revertRenameld*
- Exclure l'effet de l'opération *rename*

Intuition

1. *id* fait partie des identifiants renommés : doit retourner son ancienne valeur

Annuler l'effet d'une opération *rename*

Ajout d'un nouveau mécanisme de transformation

- Prend la forme de l'algorithme *revertRenameId*
- Exclure l'effet de l'opération *rename*

Intuition

1. *id* fait partie des identifiants renommés : doit retourner son ancienne valeur
2. *id* a (potentiellement) été inséré en concurrence : doit restaurer sa (potentielle) ancienne valeur

Annuler l'effet d'une opération *rename*

Ajout d'un nouveau mécanisme de transformation

- Prend la forme de l'algorithme *revertRenamedId*
- Exclure l'effet de l'opération *rename*

Intuition

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

Annuler l'effet d'une opération *rename*

Ajout d'un nouveau mécanisme de transformation

- Prend la forme de l'algorithme *revertRenamed*
- Exclure l'effet de l'opération *rename*

Intuition

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

Distingue cas par filtrage par motif

Annuler l'effet d'une opération *rename*

Ajout d'un nouveau mécanisme de transformation

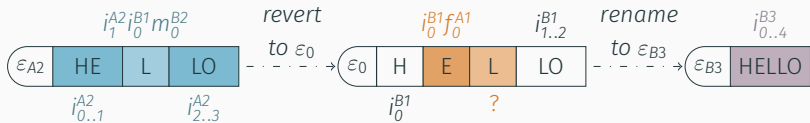
- Prend la forme de l'algorithme *revertRenamed*
- Exclure l'effet de l'opération *rename*

Intuition

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

Distingue cas par filtrage par motif

Exemple de `revertRenameId`

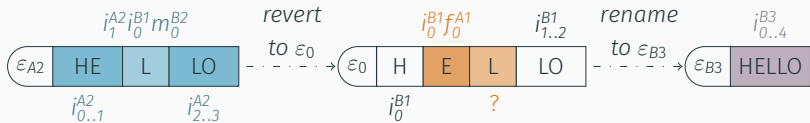


Rappel :

$$\text{renIds}_{A2} = [i_0^{B1}, i_0^{B1} f_0^{A1}, i_1^{B1}, i_2^{B1}]$$

Exemple avec $i_1^{A2} i_0^{B1} m_0^{B2}$

Exemple de `revertRenameId`



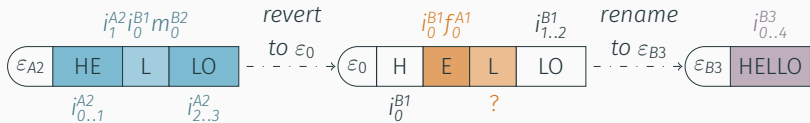
Rappel :

$$\text{renIds}_{A2} = [i_0^{B1}, i_0^{B1} f_0^{A1}, i_1^{B1}, i_2^{B1}]$$

Exemple avec $i_1^{A2} i_0^{B1} m_0^{B2}$

- Est de la forme $i_1^{A2} \oplus i_0^{B1} m_0^{B2}$: cas 2 ou 3

Exemple de `revertRenameId`



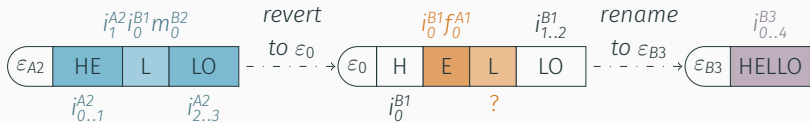
Rappel :

$$renIds_{A2} = [i_0^{B1}, i_0^{B1} f_0^{A1}, i_1^{B1}, i_2^{B1}]$$

Exemple avec $i_1^{A2} i_0^{B1} m_0^{B2}$

- Est de la forme $i_1^{A2} \oplus i_0^{B1} m_0^{B2}$: cas 2 ou 3
- Trouver l'équivalent de i_1^{A2} dans $renIds_{A2}$: $i_0^{B1} f_0^{A1}$

Exemple de `revertRenameId`



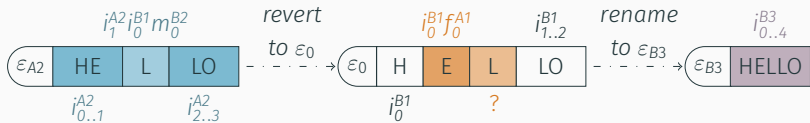
Rappel :

$$renIds_{A2} = [i_0^{B1}, i_0^{B1} f_0^{A1}, i_1^{B1}, i_2^{B1}]$$

Exemple avec $i_1^{A2} i_0^{B1} m_0^{B2}$

- Est de la forme $i_1^{A2} \oplus i_0^{B1} m_0^{B2}$: cas 2 ou 3
- Trouver l'équivalent de i_1^{A2} dans $renIds_{A2}$: $i_0^{B1} f_0^{A1}$
- Trouver l'équivalent de i_2^{A2} dans $renIds_{A2}$: i_1^{B1}

Exemple de revertRenameId



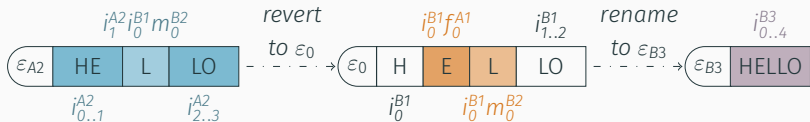
Rappel :

$$\text{renIds}_{A2} = [i_0^{B1}, i_0^{B1} f_0^{A1}, i_1^{B1}, i_2^{B1}]$$

Exemple avec $i_1^{A2} i_0^{B1} m_0^{B2}$

- Est de la forme $i_1^{A2} \oplus i_0^{B1} m_0^{B2}$: cas 2 ou 3
- Trouver l'équivalent de i_1^{A2} dans renIds_{A2} : $i_0^{B1} f_0^{A1}$
- Trouver l'équivalent de i_2^{A2} dans renIds_{A2} : i_1^{B1}
- Comparer $i_0^{B1} m_0^{B2}$ avec ces derniers : $i_0^{B1} f_0^{A1} <_{id} i_0^{B1} m_0^{B2} <_{id} i_1^{B1}$

Exemple de revertRenameId



Rappel :

$$renIds_{A2} = [i_0^{B1}, i_0^{B1}f_0^{A1}, i_1^{B1}, i_2^{B1}]$$

Exemple avec $i_1^{A2}i_0^{B1}m_0^{B2}$

- Est de la forme $i_1^{A2} \oplus i_0^{B1}m_0^{B2}$: cas 2 ou 3
- Trouver l'équivalent de i_1^{A2} dans $renIds_{A2}$: $i_0^{B1}f_0^{A1}$
- Trouver l'équivalent de i_2^{A2} dans $renIds_{A2}$: i_1^{B1}
- Comparer $i_0^{B1}m_0^{B2}$ avec ces derniers : $i_0^{B1}f_0^{A1} <_{id} i_0^{B1}m_0^{B2} <_{id} i_1^{B1}$
- Retourner $i_0^{B1}m_0^{B2}$

RenamableLogootSplit

Validation

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

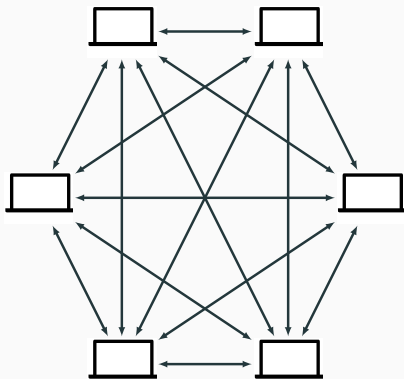
- Montrer convergence des noeuds
- Montrer que mécanisme de renommage améliore performances de la séquence répliquée (mémoire, calculs, bande-passante)

Conduite d'une évaluation expérimentale

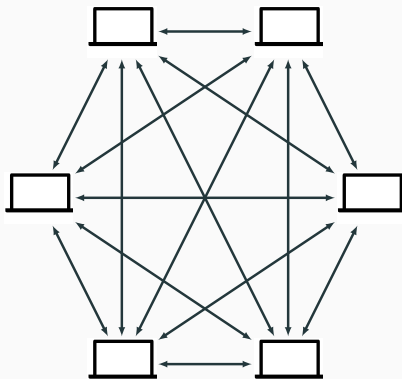
Absence d'un jeu de données de sessions
d'édition collaborative

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

Mise en place de simulations pour générer un
jeu de données



- 10 noeuds éditent collaborativement un document
- Utilisent soit LogootSplit (LS), soit RenamableLogootSplit (RLS)



- 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

- 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)

- 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)

- 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**

- 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

*. Code des simulations et benchmarks :

<https://github.com/coast-team/mute-bot-random>

- 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

Intuition

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

Intuition

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

- Ensemble des noeuds convergent

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

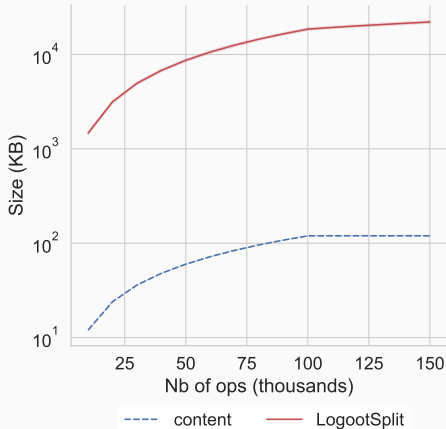
Intuition

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

Surcoût en métadonnées - 1 noeud de renommage

Intuition

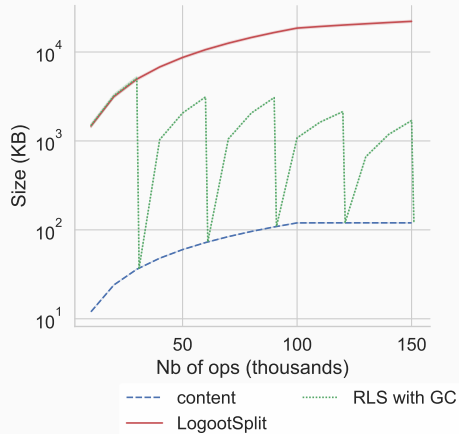
Mesurer évolution de la taille de la structure de données à partir des instantanés des sessions avec 1 seul noeud de renommage



Surcoût en métadonnées - 1 noeud de renommage

Intuition

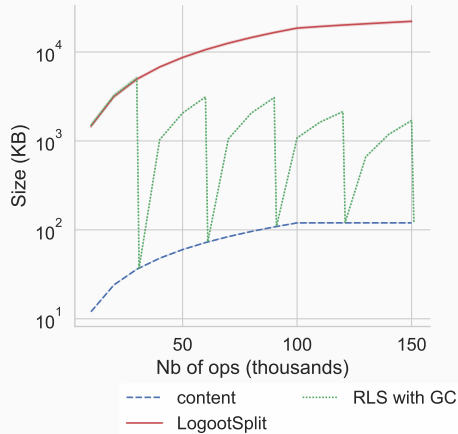
Mesurer évolution de la taille de la structure de données à partir des instantanés des sessions avec 1 seul noeud de renommage



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

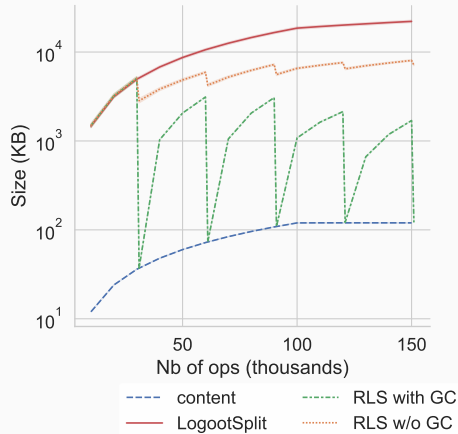


- Opération *rename* réinitialise surcoût du CRDT, si GC de l'entièreté des métadonnées du mécanisme de renommage

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

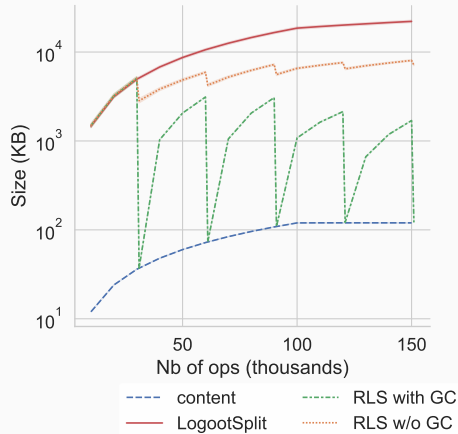


- Opération *rename* réinitialise surcoût du CRDT, si GC de l'entièreté des métadonnées du mécanisme de renommage

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

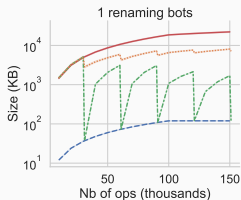


- Opération *rename* réinitialise surcoût du CRDT, si GC de l'entièreté des métadonnées du mécanisme de renommage
- Opération *rename* réduit de 66% surcoût du CRDT sinon

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

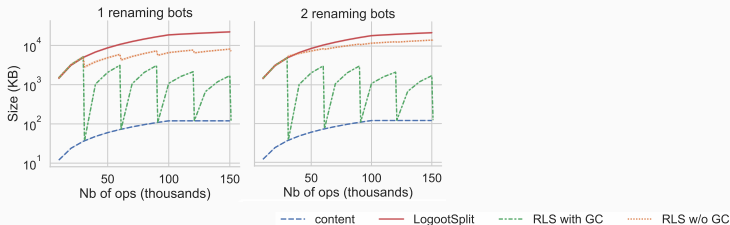


--- content — LogootSplit - - - RLS with GC RLS w/o GC

Surcoût en métadonnées - 1 à 4 noeuds de renommage

Intuition

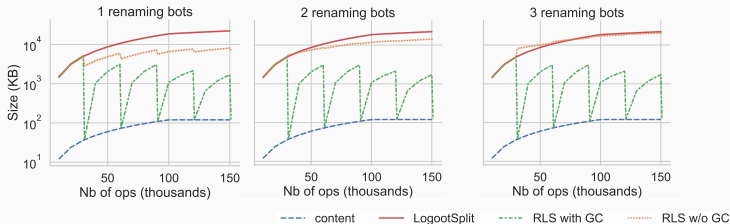
Mesurer évolution de la taille de la structure de données en fonction du nombre de noeuds de renommage



Surcoût en métadonnées - 1 à 4 noeuds de renommage

Intuition

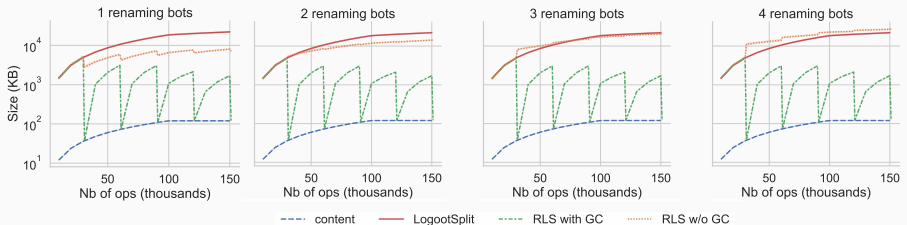
Mesurer évolution de la taille de la structure de données en fonction du nombre de noeuds de renommage



Surcoût en métadonnées - 1 à 4 noeuds de renommage

Intuition

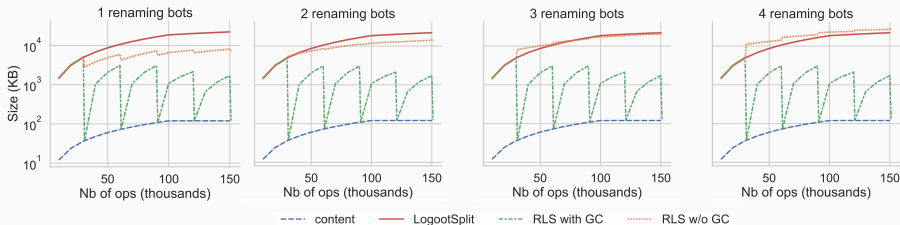
Mesurer évolution de la taille de la structure de données en fonction du nombre de noeuds de renommage



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**



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

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

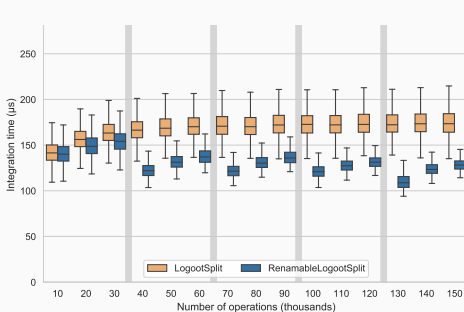
Intuition

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

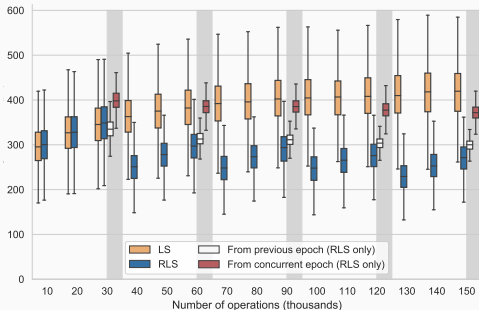
Surcoût en calculs - Opérations *insert* et *remove*

Intuition

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



(a) Modifications locales

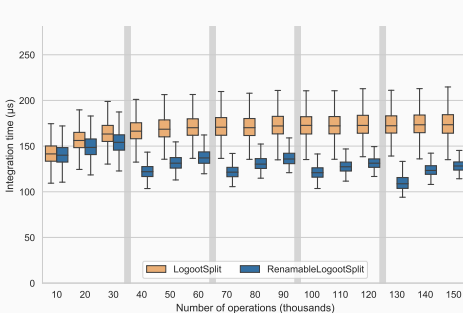


(b) Modifications distantes

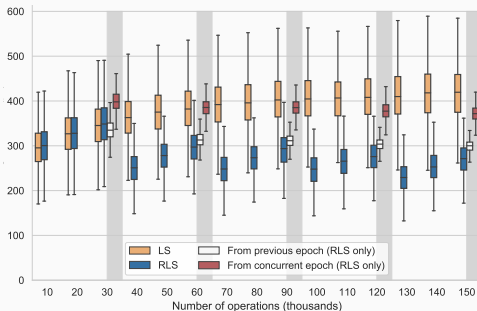
Surcoût en calculs - Opérations *insert* et *remove*

Intuition

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



(a) Modifications locales



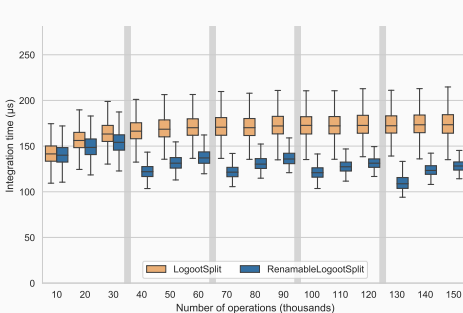
(b) Modifications distantes

- Opérations *rename* réduisent temps intégration

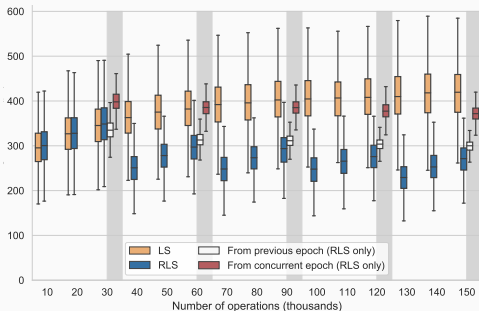
Surcoût en calculs - Opérations *insert* et *remove*

Intuition

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



(a) Modifications locales



(b) Modifications distantes

- Opérations *rename* réduisent temps intégration
- Réduction état contrebalance surcoût transformation

Surcoûts en calculs - Opérations *rename*

Intuition

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

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	1 ^{er} Percent.	99 ^{ème} 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
Opération <i>rename</i> distante même époque	30	481	477	15.2	454	537
	90	1491	1482	58.8	1396	1658
	150	1694	1676	60.6	1591	1853
Opération <i>rename</i> distante plus prioritaire	30	644	644	16.6	620	683
	90	1998	1994	46.6	1906	2112
	150	2242	2234	63.5	2139	2351

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	1 ^{er} Percent.	99 ^{ème} 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
Opération <i>rename</i> distante même époque	30	481	477	15.2	454	537
	90	1491	1482	58.8	1396	1658
	150	1694	1676	60.6	1591	1853
Opération <i>rename</i> distante 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

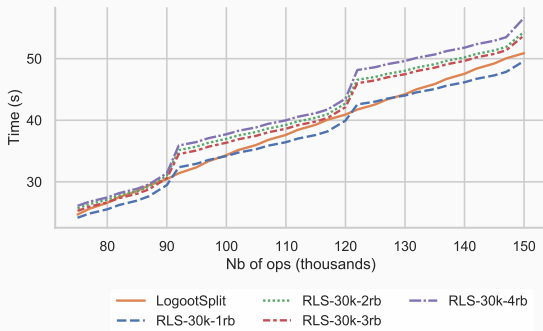
Intuition

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

Surcoût en calculs - Vue globale

Intuition

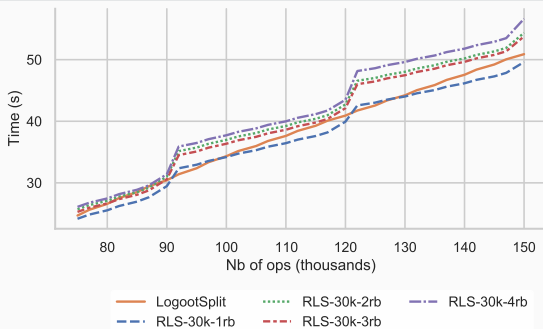
Mesurer temps pour intégrer l'entièreté du journal d'opérations d'une collaboration en fonction du nombre de noeuds de renommage



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**



- Initialement, gains sur opérations *insert* et *remove* contrebalancent coût des opérations *rename* ...
- ...mais coût des opérations *rename* augmente surcoût total *in fine*

RenamableLogootSplit

Conclusion

Contribution

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

[6]. BAQUERO et al., *Pure Operation-Based Replicated Data Types*.

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^[6] requise pour supprimer les métadonnées

[6]. BAQUERO et al., *Pure Operation-Based Replicated Data Types*.

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^[6] requise pour supprimer les métadonnées

Perspectives

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

[6]. BAQUERO et al., *Pure Operation-Based Replicated Data Types*.

Conclusion générale & Perspectives

Contributions

- Conception d'un mécanisme de renommage pour CRDTs pour le type Séquence à identifiants densément ordonnés
 - Implémentation et instrumentation de RenamableLogootSplit et de ses dépendances (protocole d'appartenance au réseau, couche de livraison)

Contributions

- Conception d'un mécanisme de renommage pour CRDTs pour le type Séquence à identifiants densément ordonnés
 - Implémentation et instrumentation de RenamableLogootSplit et de ses dépendances (protocole d'appartenance au réseau, couche de livraison)
- Comparaison des différents modèles de synchronisation pour CRDTs...
- ...et des différentes approches pour CRDTs pour le type Séquence

Limites & perspectives

Limites de RenamableLogootSplit

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

Perspectives autour de RenamableLogootSplit

- Comment définir une relation $priority <_{\epsilon}$ réduisant calculs à échelle du système ?
- Est-ce que RenamableLogootSplit est correct ?

Limites de RenamableLogootSplit

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

Perspectives autour de RenamableLogootSplit

- Comment définir une relation $priority <_{\epsilon}$ réduisant calculs à échelle du système?
- Est-ce que RenamableLogootSplit est correct?

Perspectives autour des CRDTs

- Doit-on encore concevoir CRDTs synchronisés par états ou opérations?
- Peut-on proposer un framework pour conception de CRDTs synchronisés par opérations?

Merci de votre attention, avez-vous des questions?



- **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>

Doit-on encore concevoir CRDTs synchronisés par états ou opérations?

	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?