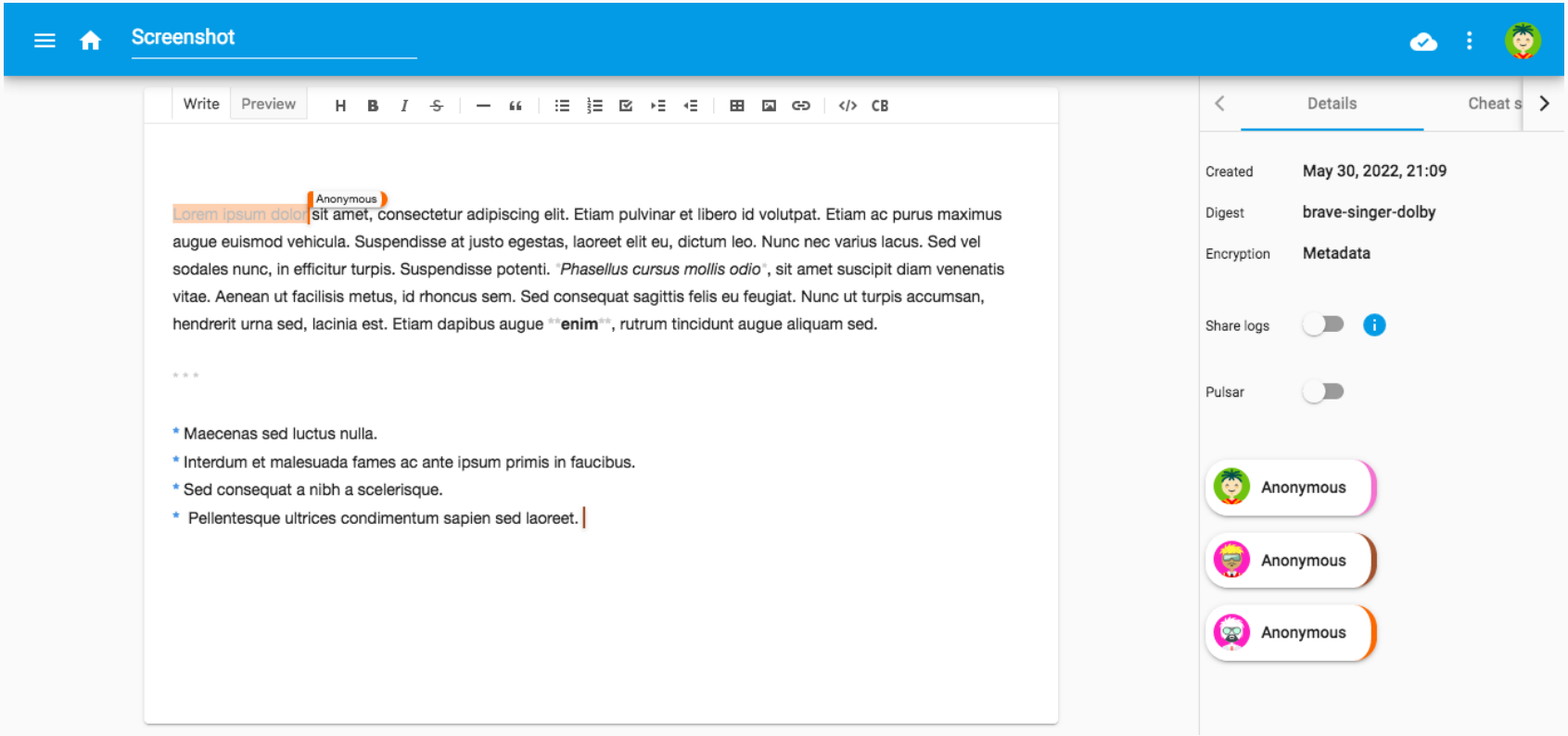
Ré-identification sans coordination dans les types de données répliquées sans conflits

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	Professeure des Universités, 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

MUTE *, un exemple de Local-First Software (LFS) [1]

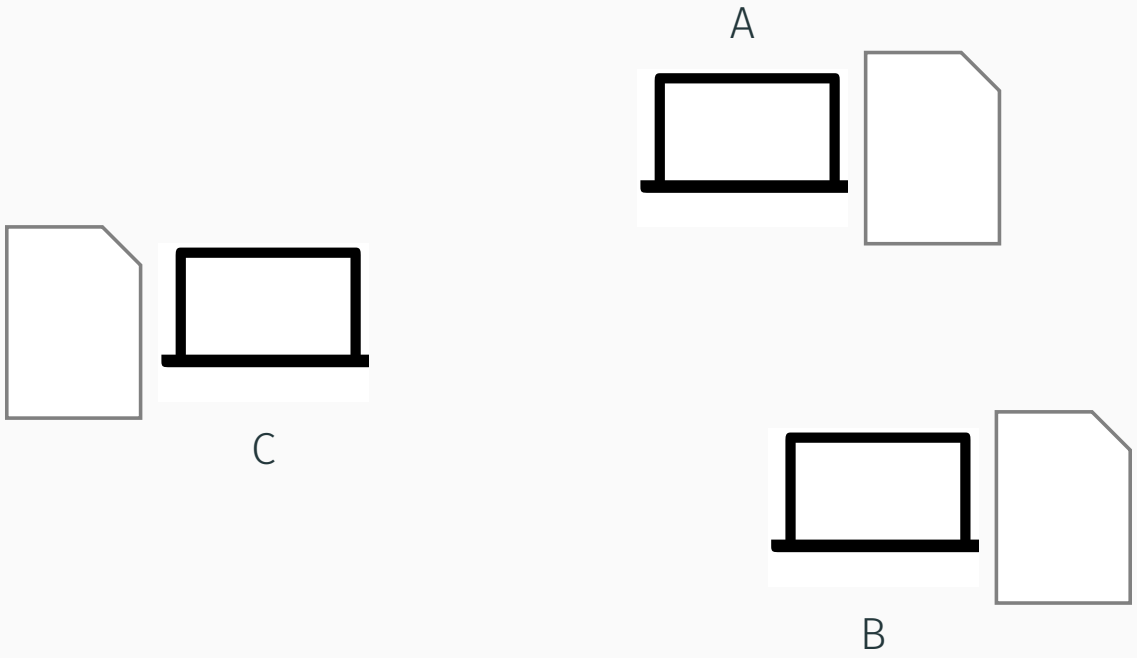


- Application pair-à-pair
- Permet de rédiger collaborativement des documents texte
- Garantit la confidentialité & souveraineté des données

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

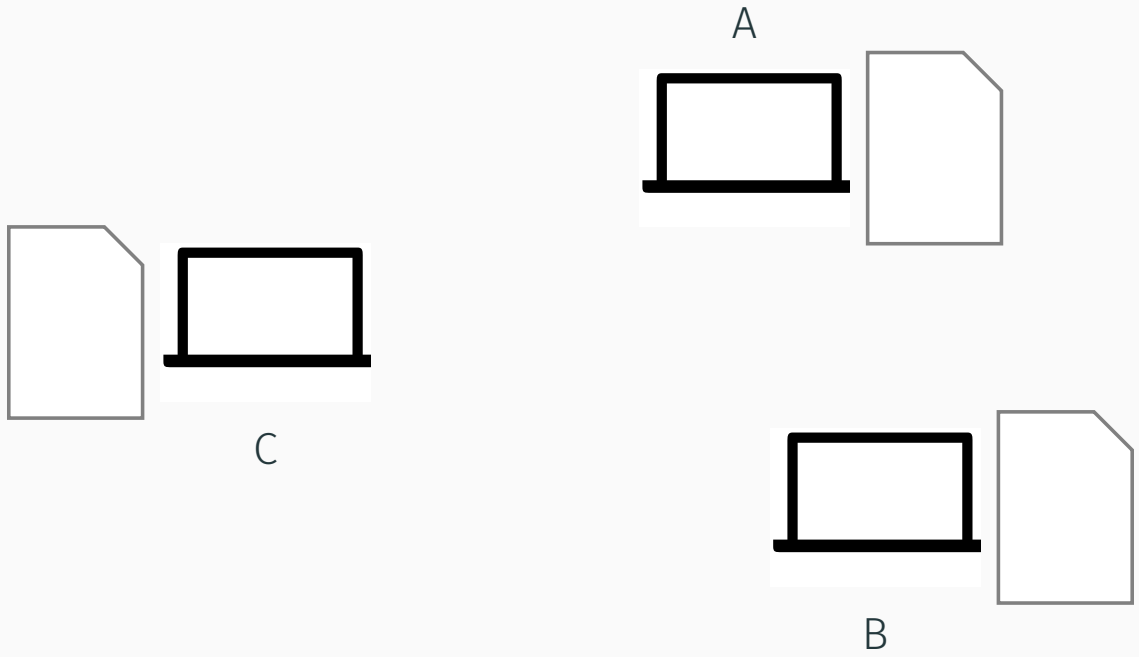
[1]. KLEPPMANN et al., « Local-First Software : You Own Your Data, in Spite of the Cloud ».

Réplication dans applications collaboratives pair-à-pair



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

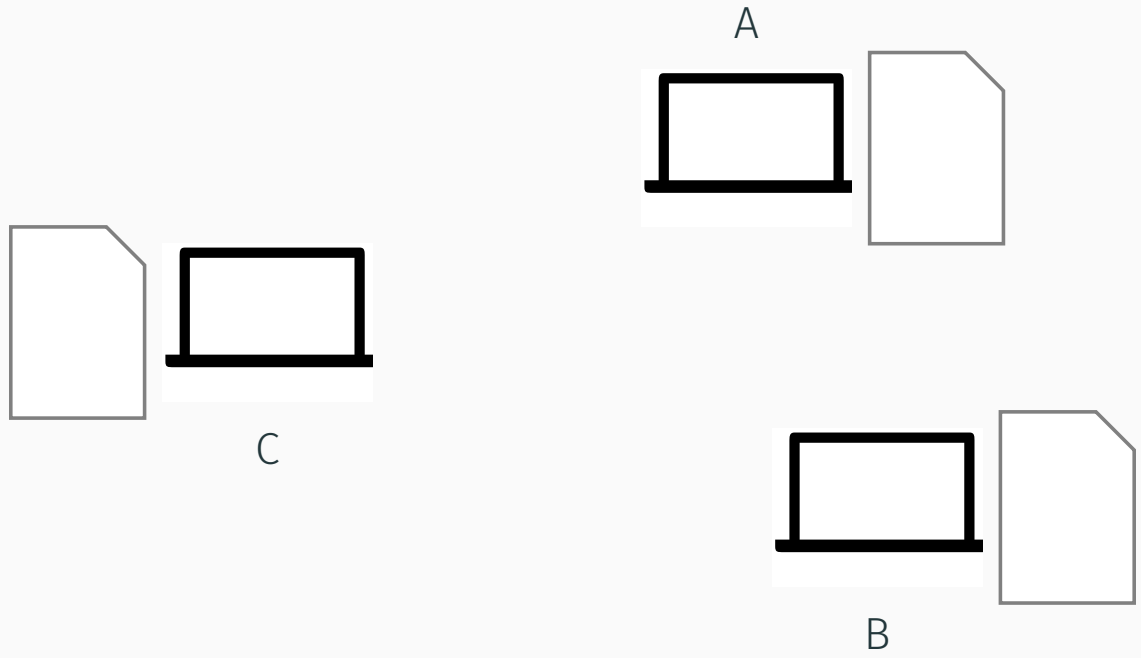
Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être déconnectés

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

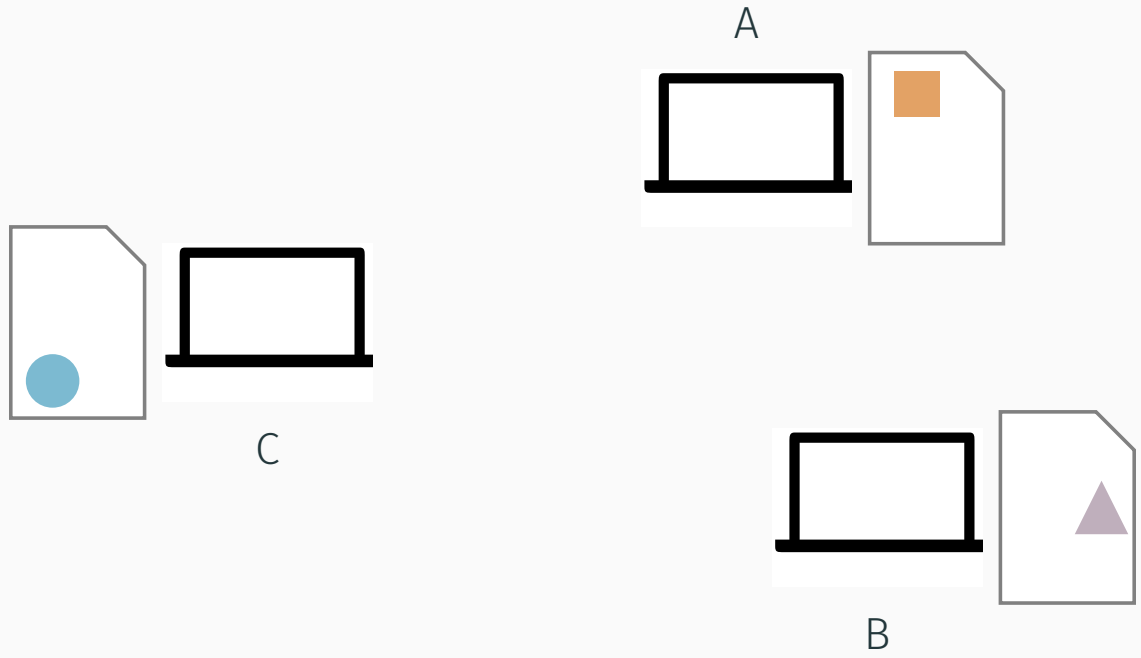
Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)

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

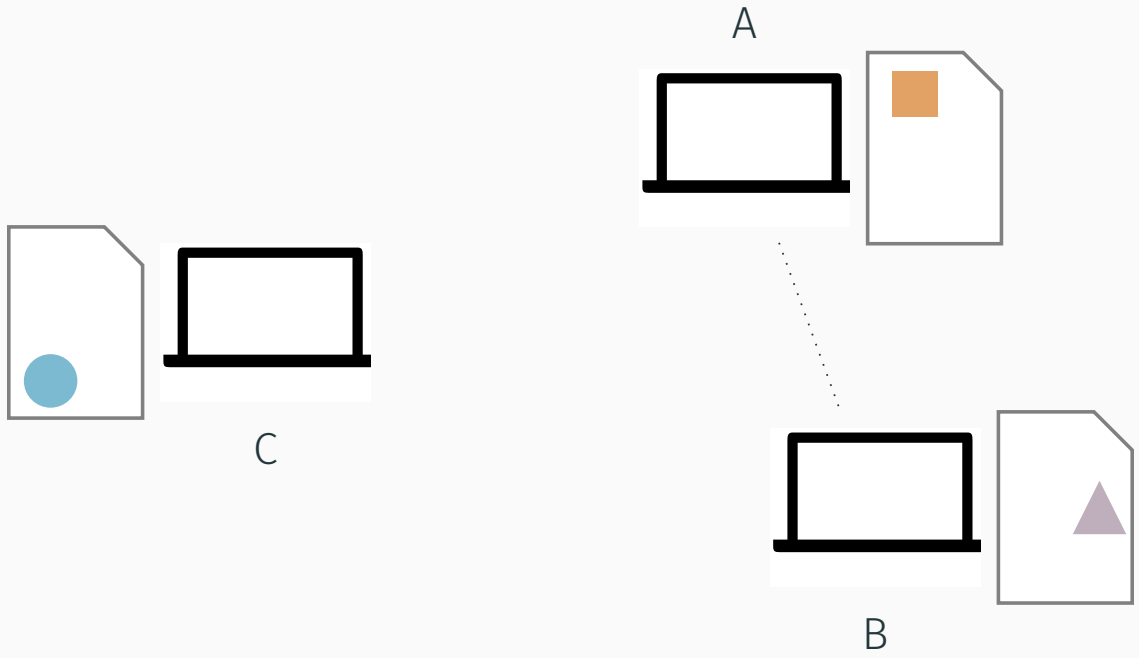
Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)

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

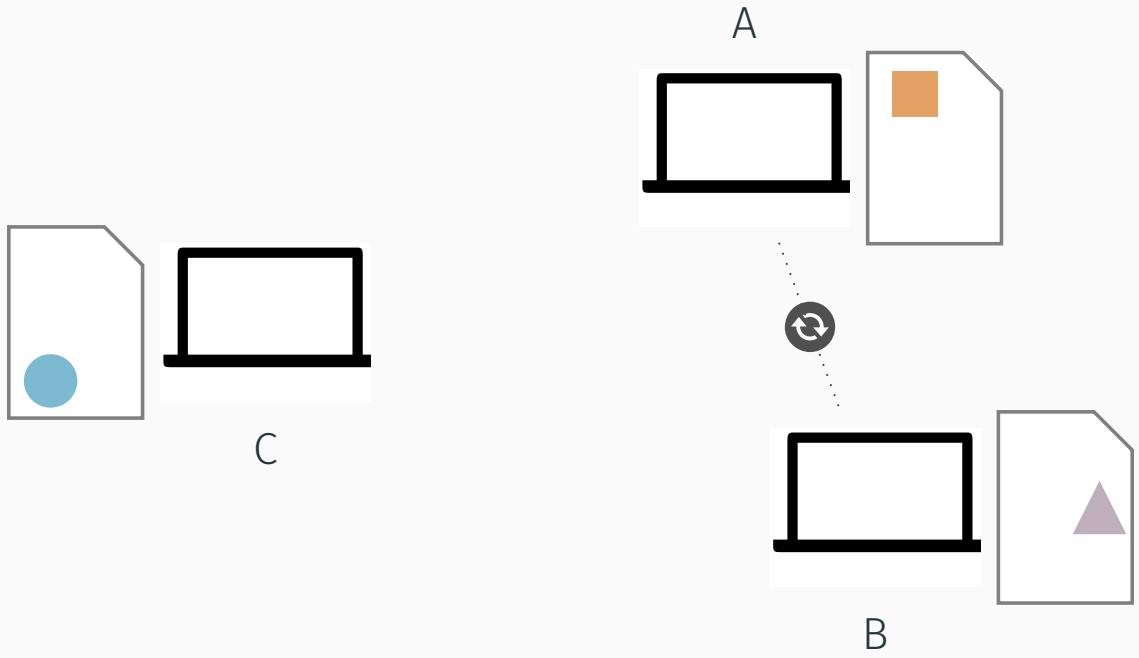
Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)

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

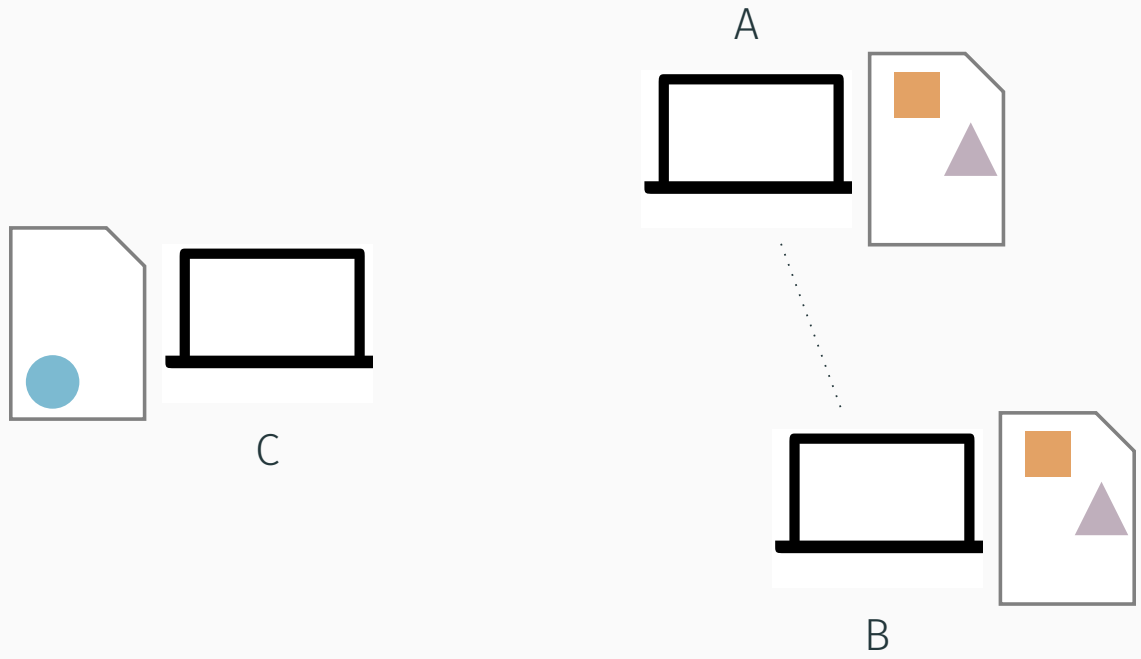
Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)

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

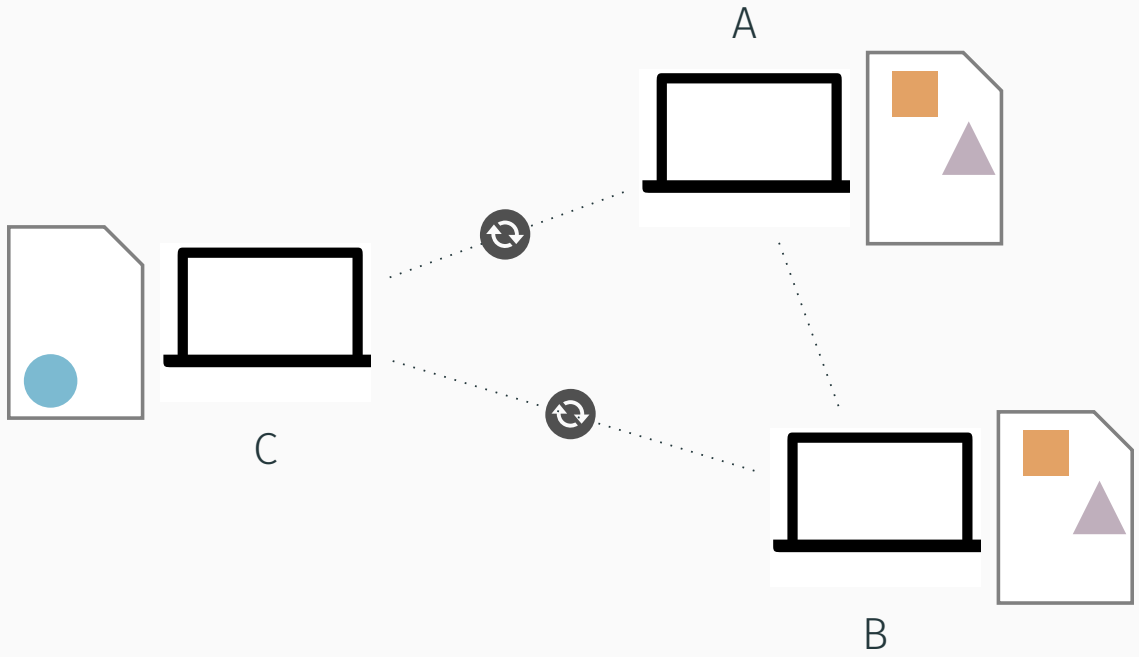
Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)

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

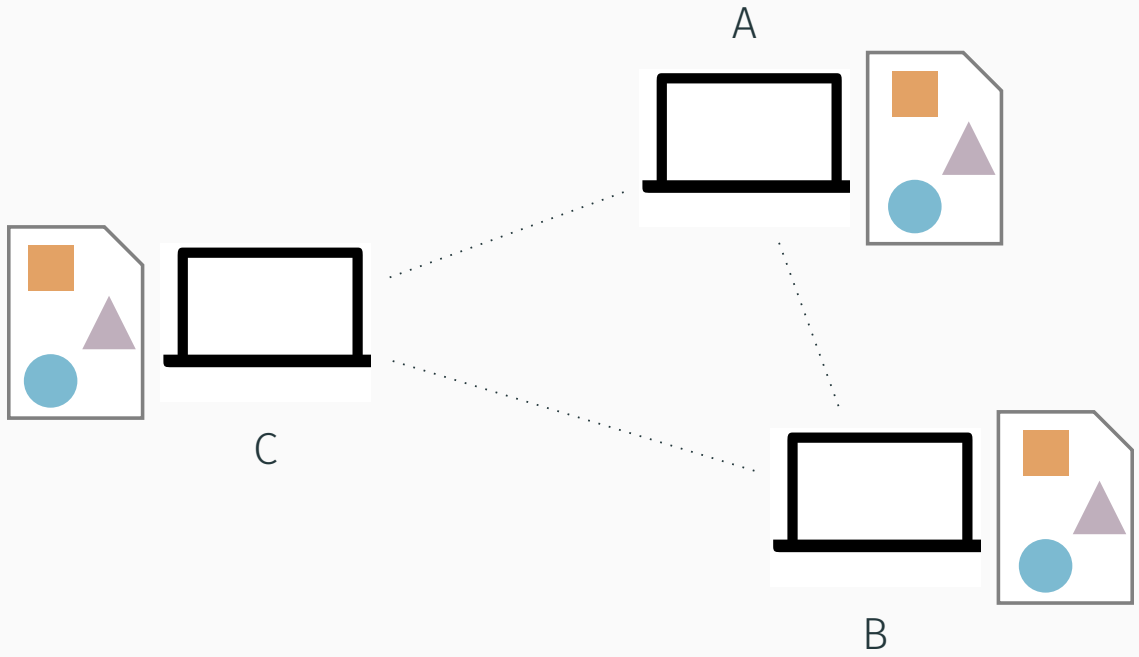
Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)

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

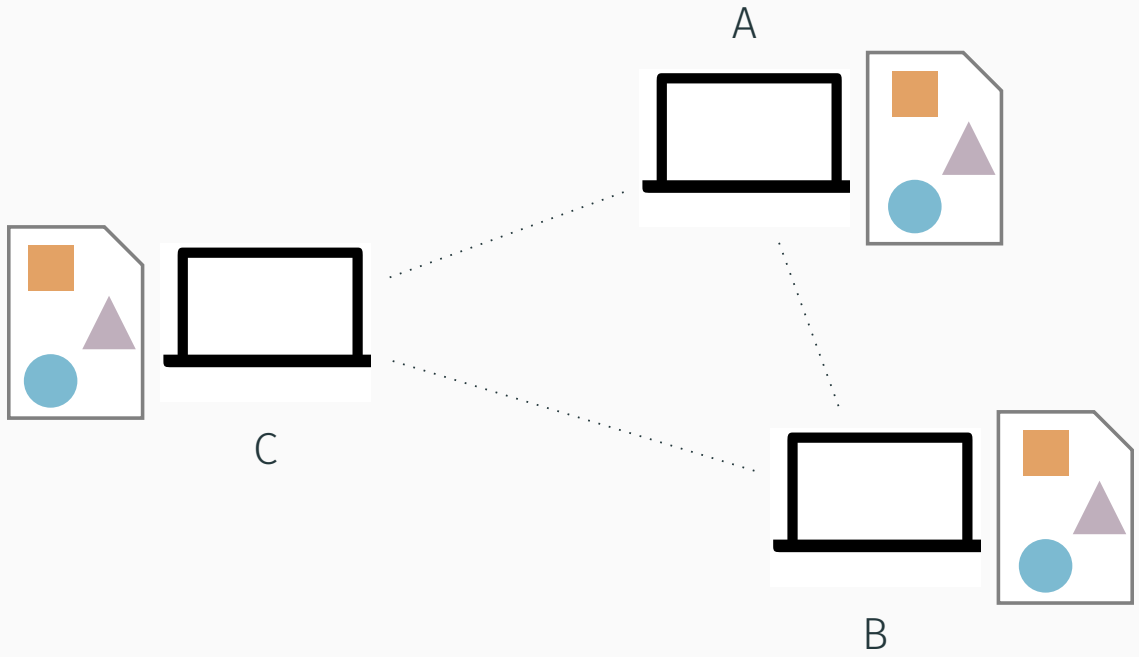
Réplication dans applications collaboratives pair-à-pair



- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)
- Doit garantir convergence à terme [1]...
- ...malgré ordres différents d'intégration des modifications

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

Réplication dans applications collaboratives pair-à-pair



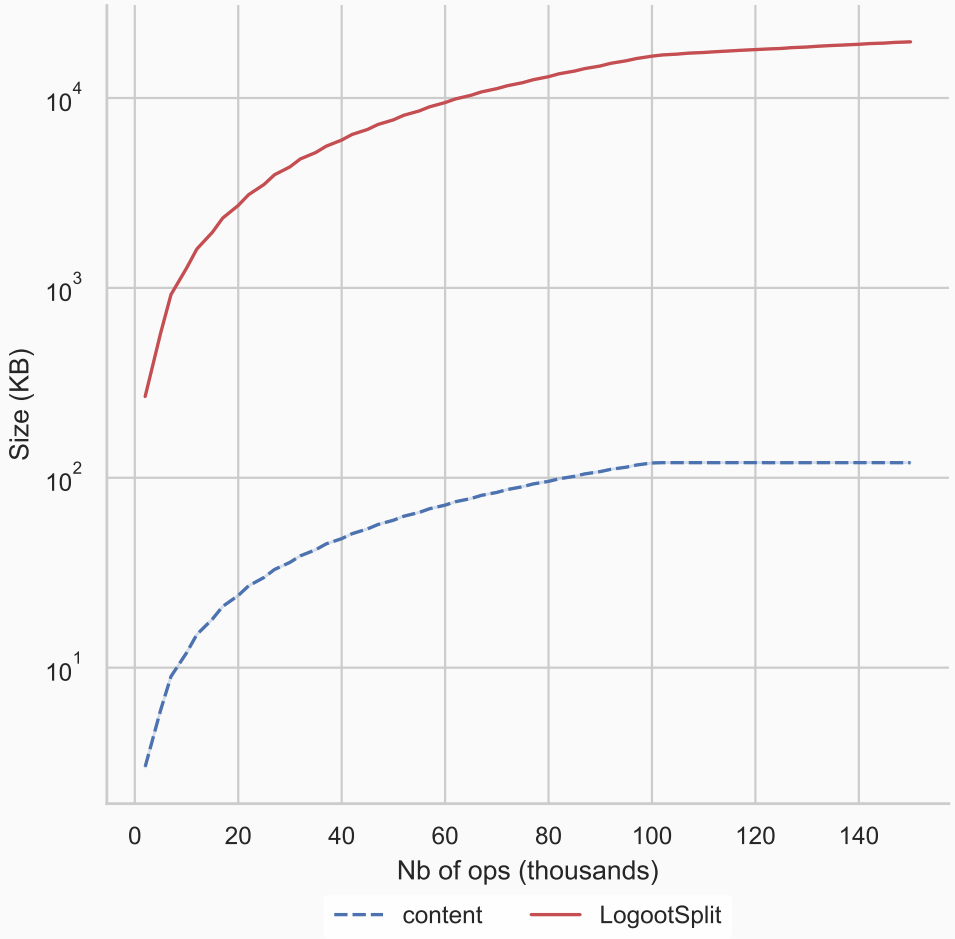
- Noeuds peuvent être déconnectés
- Doivent pouvoir travailler sans coordination synchrone préalable (par ex. consensus)
- Doit garantir convergence à terme [1]...
- ...malgré ordres différents d'intégration des modifications

Nécessite des *mécanismes de résolution de conflits*

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

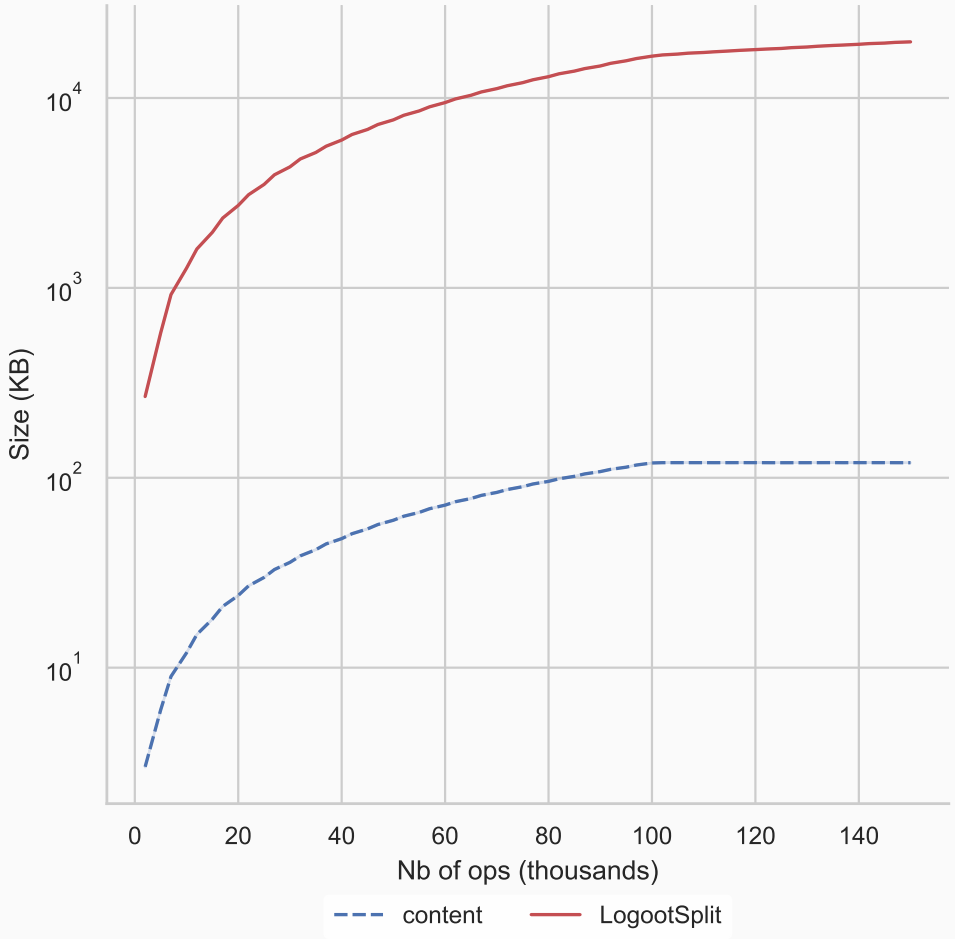
Évaluation de MUTE

Taille du texte comparée à taille de la séquence répliquée



Évaluation de MUTE

Taille du texte comparée à taille de la séquence répliquée

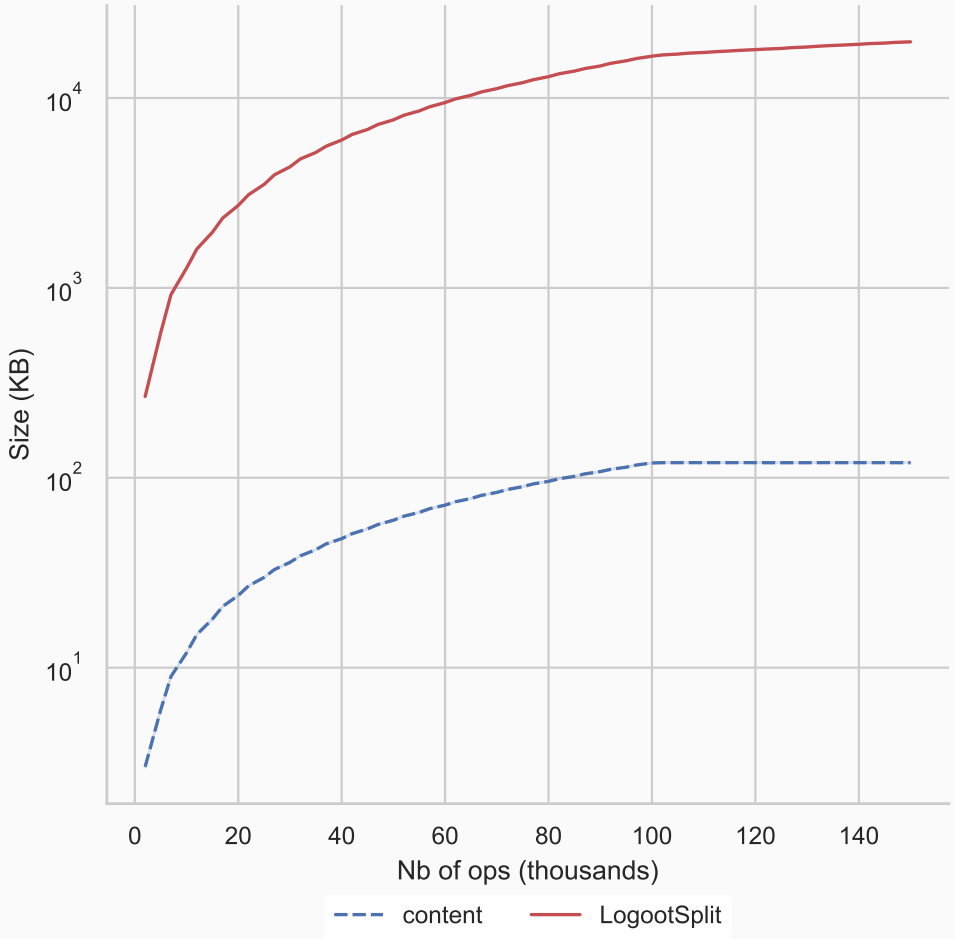


Constat

- 1% contenu...
- ...99% métadonnées

Évaluation de MUTE

Taille du texte comparée à taille de la séquence répliquée



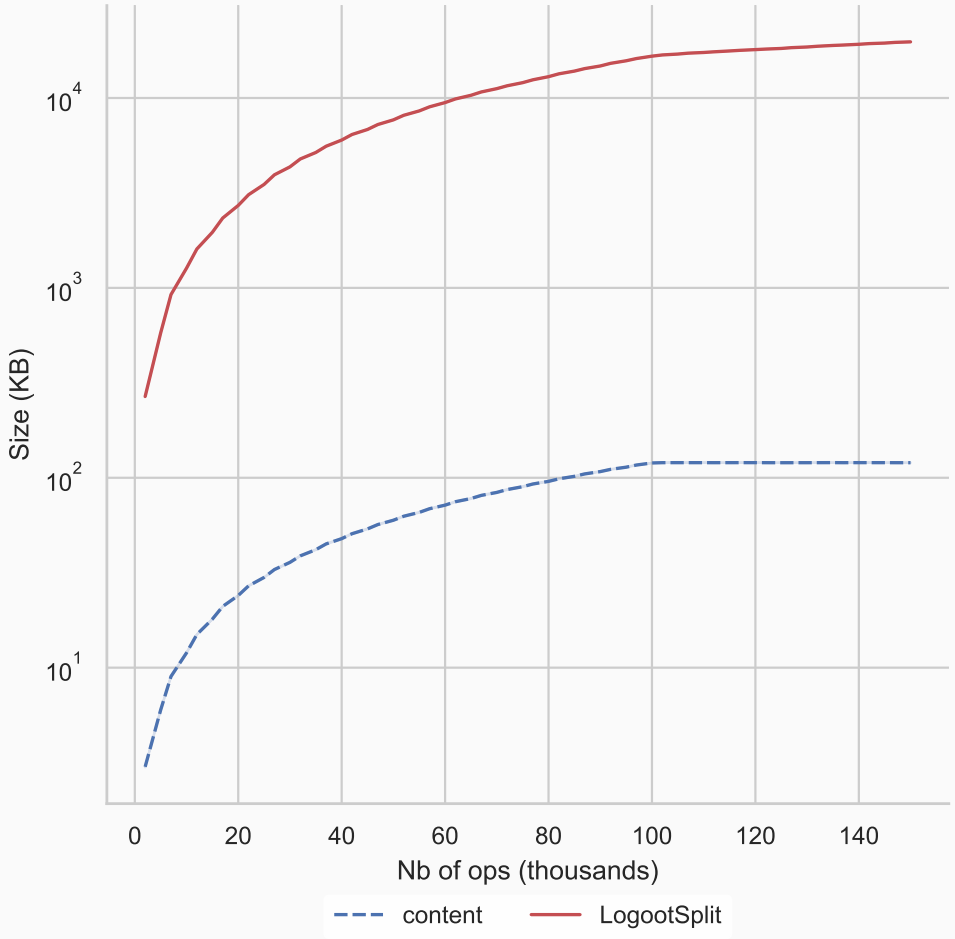
Constat

- 1% contenu...
- ...99% métadonnées

Et ça augmente !

Évaluation de MUTE

Taille du texte comparée à taille de la séquence répliquée



Constat

- 1% contenu...
- ...99% métadonnées

Et ça augmente !

Impact

- Surcoût mémoire...
- ...mais aussi surcoût en calculs et en bande-passante

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

Plan de la présentation

Plan

- L'origine de la croissance du surcoût des mécanismes de résolution de conflits pour le type Séquence

Plan de la présentation

Plan

- L'origine de la croissance du surcoût des mécanismes de résolution de conflits pour le type Séquence
- **Contribution** : Un mécanisme pair-à-pair de réduction du surcoût des mécanismes de résolution de conflits

Plan de la présentation

Plan

- L'origine de la croissance du surcoût des mécanismes de résolution de conflits pour le type Séquence
- **Contribution** : Un mécanisme pair-à-pair de réduction du surcoût des mécanismes de résolution de conflits
- Conclusion générale & perspectives

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

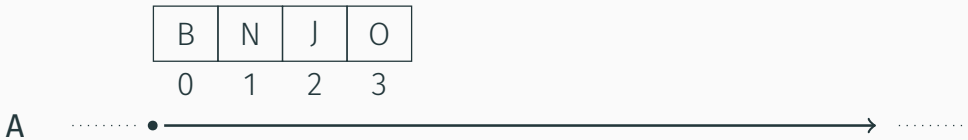
Convergence forte

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 ».

CRDTs pour le type Séquence

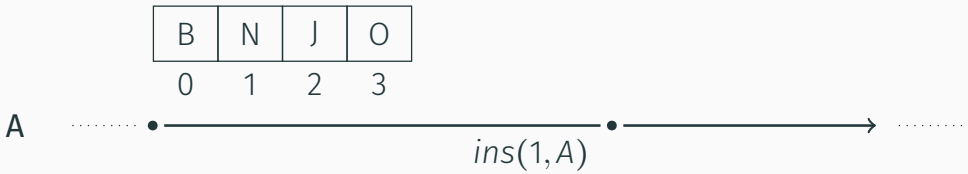
Type Séquence usuel



[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 ».

CRDTs pour le type Séquence

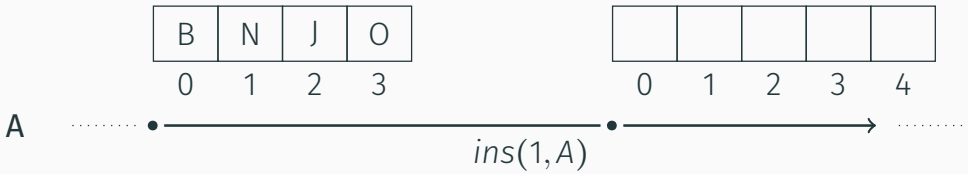
Type Séquence usuel



[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 ».

CRDTs pour le type Séquence

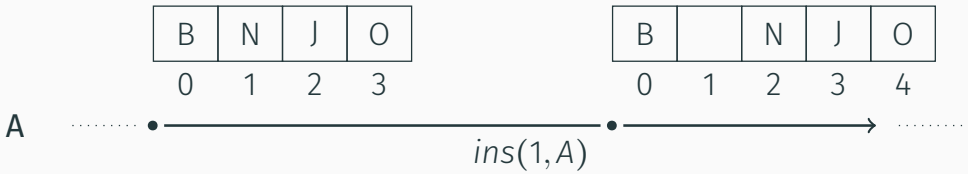
Type Séquence usuel



[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 ».

CRDTs pour le type Séquence

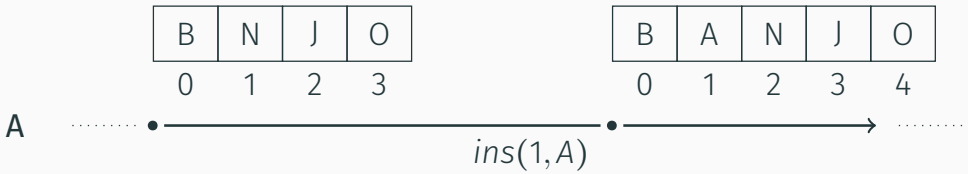
Type Séquence usuel



[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 ».

CRDTs pour le type Séquence

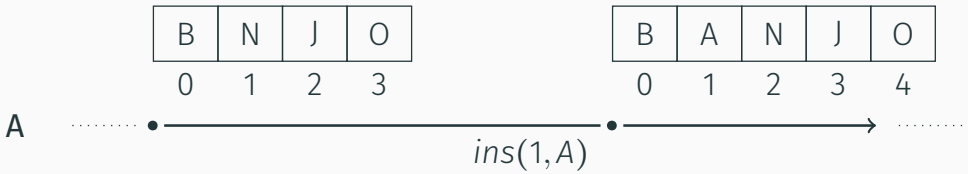
Type Séquence usuel



[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 ».

CRDTs pour le type Séquence

Type Séquence usuel

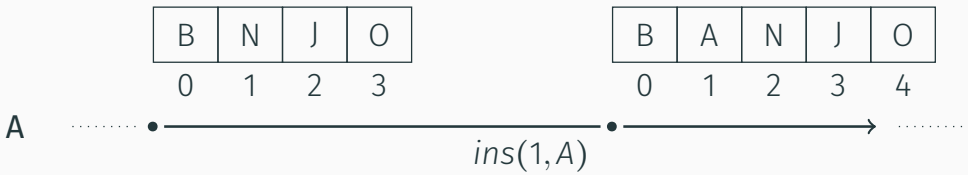


- Changements des indices est source de conflits

[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 ».

CRDTs pour le type Séquence

Type Séquence usuel



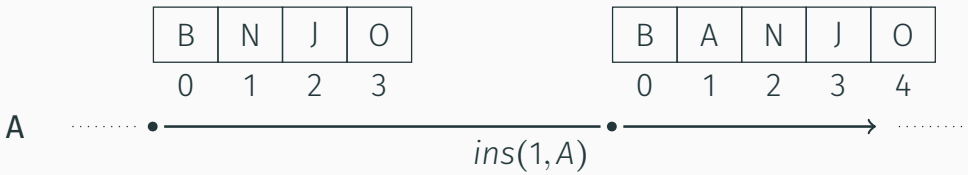
CRDTs pour Séquence

- Changements des indices est source de conflits
- Assignent des identifiants de position [3] à chaque élément
- Permettent d'ordonner les élements

[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 ».

CRDTs pour le type Séquence

Type Séquence usuel



CRDTs pour Séquence

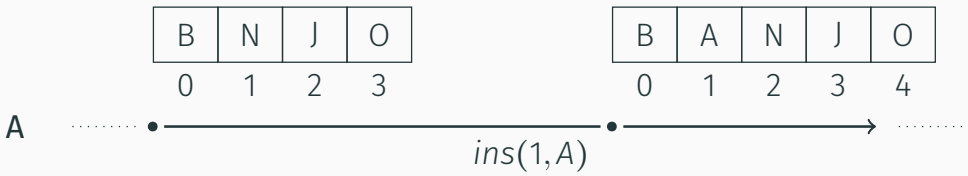
- Changements des indices est source de conflits
- Assignent des identifiants de position [3] à chaque élément
- Permettent d'ordonner les élements

$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

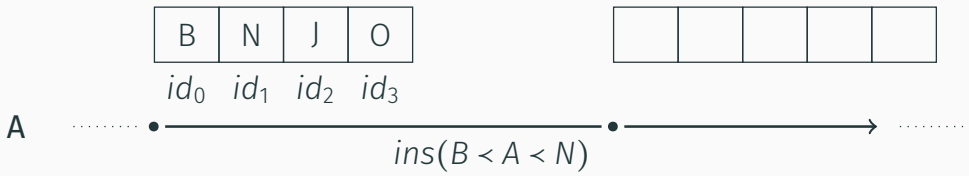
[3]. Preguica et al., « A Commutative Replicated Data Type for Cooperative Editing ».
[4]. André et al., « Supporting Adaptable Granularity of Changes for Massive-Scale Collaborative Editing ».

CRDTs pour le type Séquence

Type Séquence usuel



CRDTs pour Séquence



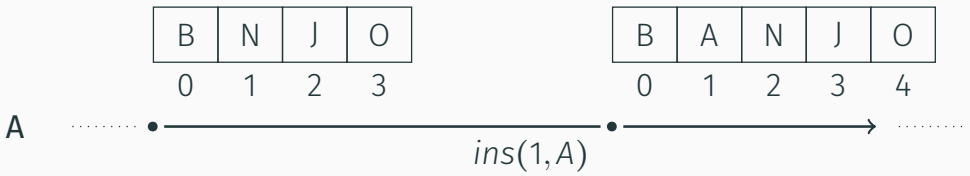
- Changements des indices est source de conflits
- Assignent des identifiants de position [3] à chaque élément
- Permettent d'ordonner les élements

$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

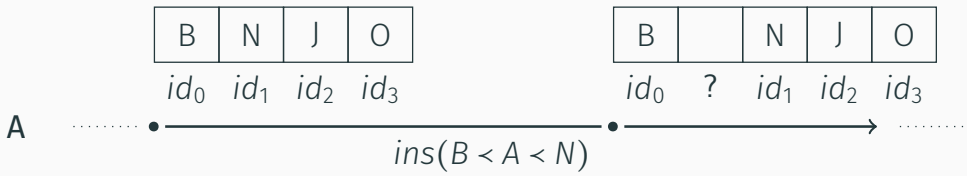
[3]. Preguica et al., « A Commutative Replicated Data Type for Cooperative Editing ».
[4]. André et al., « Supporting Adaptable Granularity of Changes for Massive-Scale Collaborative Editing ».

CRDTs pour le type Séquence

Type Séquence usuel



CRDTs pour Séquence



- Changements des indices est source de conflits
- Assignent des identifiants de position [3] à chaque élément
- Permettent d'ordonner les élements

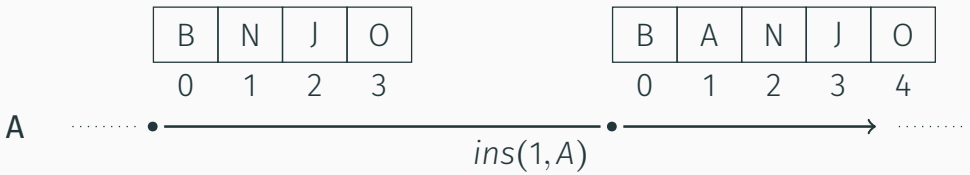
$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

- Appartiennent à un espace dense

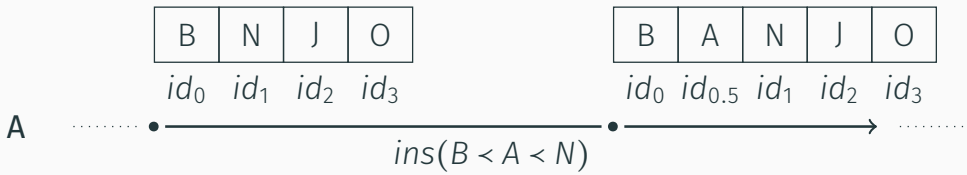
[3]. Preguica et al., « A Commutative Replicated Data Type for Cooperative Editing ».
[4]. André et al., « Supporting Adaptable Granularity of Changes for Massive-Scale Collaborative Editing ».

CRDTs pour le type Séquence

Type Séquence usuel



CRDTs pour Séquence



- Changements des indices est source de conflits
- Assignent des identifiants de position [3] à chaque élément
- Permettent d'ordonner les élements

$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

- Appartiennent à un espace dense

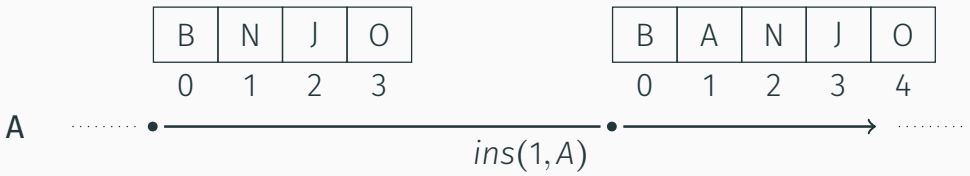
$$id_0 <_{id} id_{0.5} <_{id} id_1$$

---

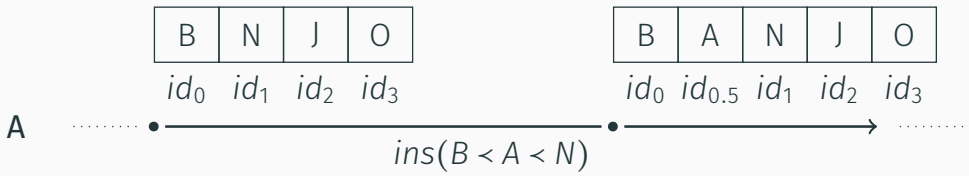
[3]. Preguica et al., « A Commutative Replicated Data Type for Cooperative Editing ».
[4]. André et al., « Supporting Adaptable Granularity of Changes for Massive-Scale Collaborative Editing ».

CRDTs pour le type Séquence

Type Séquence usuel



CRDTs pour Séquence



- Changements des indices est source de conflits
- Assignent des identifiants de position [3] à chaque élément
- Permettent d'ordonner les élements

$$id_0 <_{id} id_1 <_{id} id_2 <_{id} id_3$$

- Appartiennent à un espace dense

$$id_0 <_{id} id_{0.5} <_{id} id_1$$

Utilise LogootSplit [4] comme base

[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 LogootSplit

Identifiant

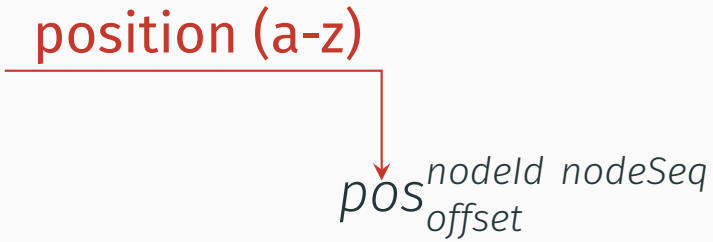
- Composé d'un ou plusieurs tuples de la forme

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

Identifiant LogootSplit

Identifiant

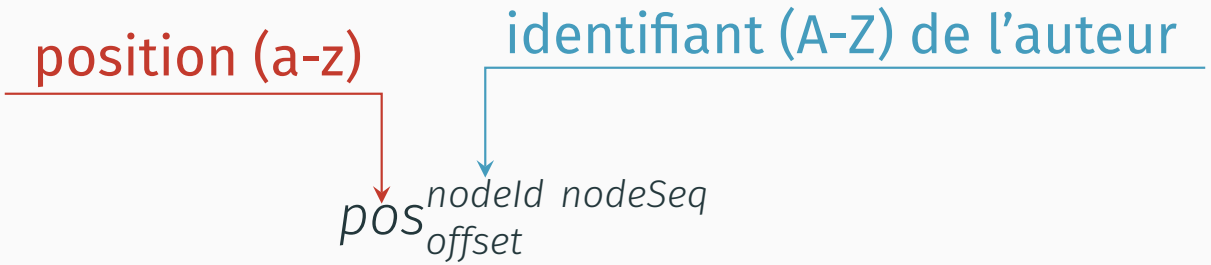
- Composé d'un ou plusieurs tuples de la forme



Identifiant LogootSplit

Identifiant

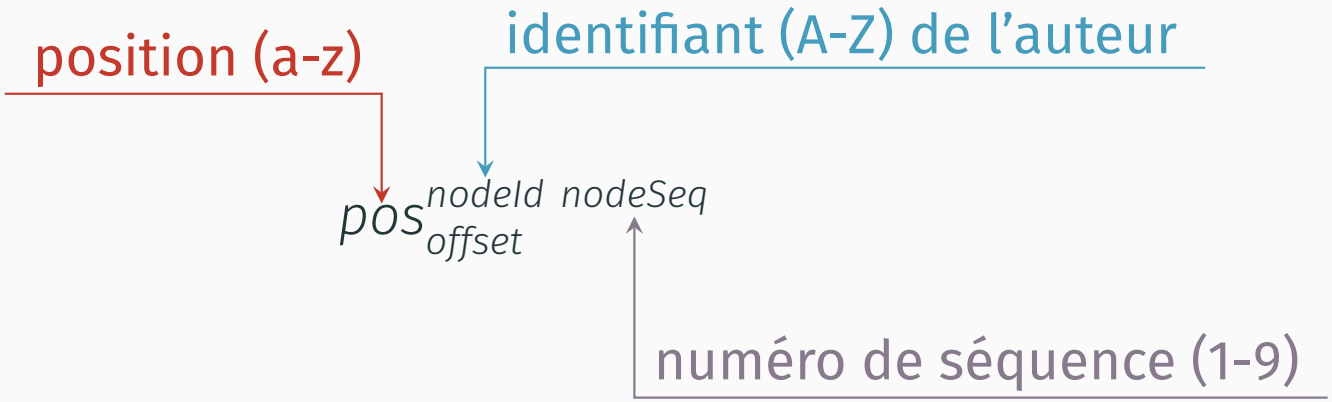
- Composé d'un ou plusieurs tuples de la forme



Identifiant LogootSplit

Identifiant

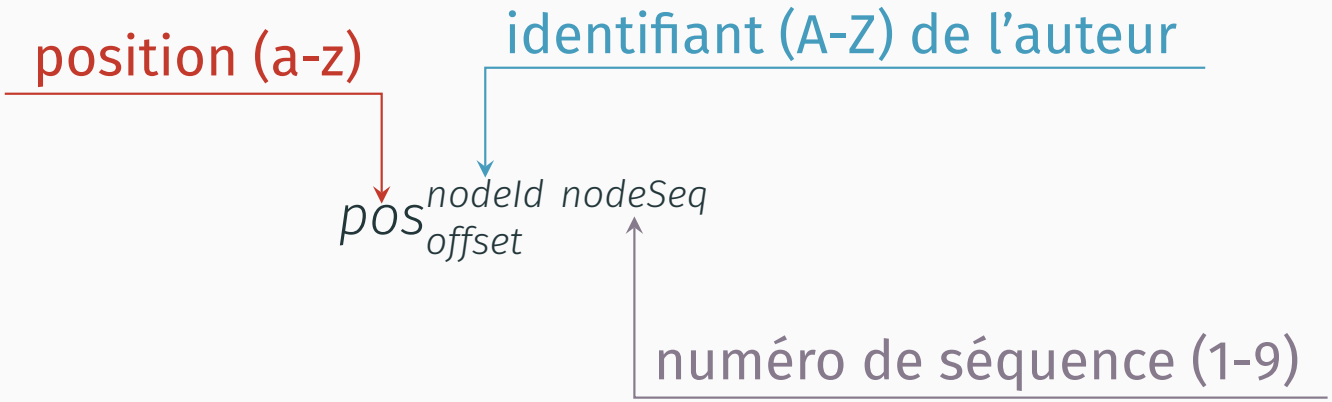
- Composé d'un ou plusieurs tuples de la forme



Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



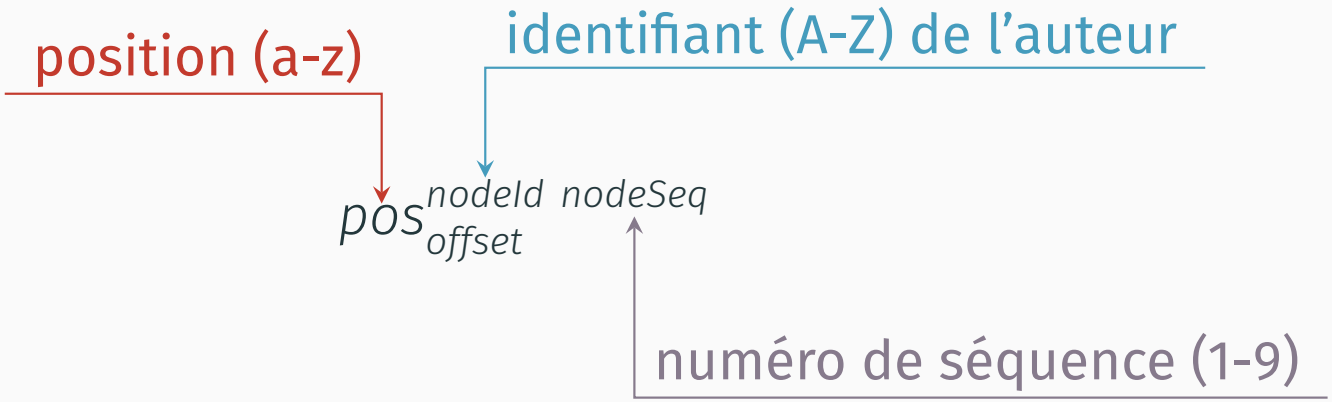
Exemples

d_0^{F5}

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



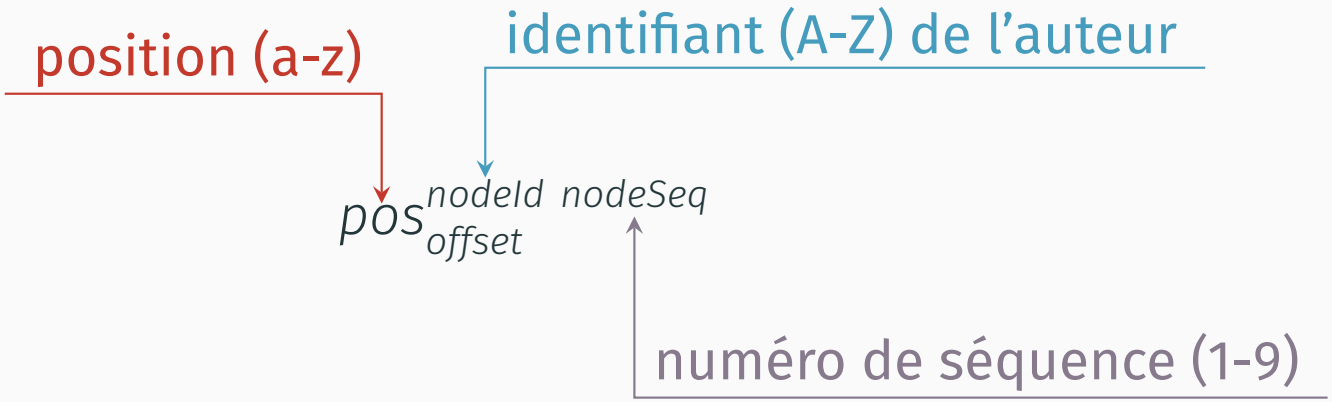
Exemples

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

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



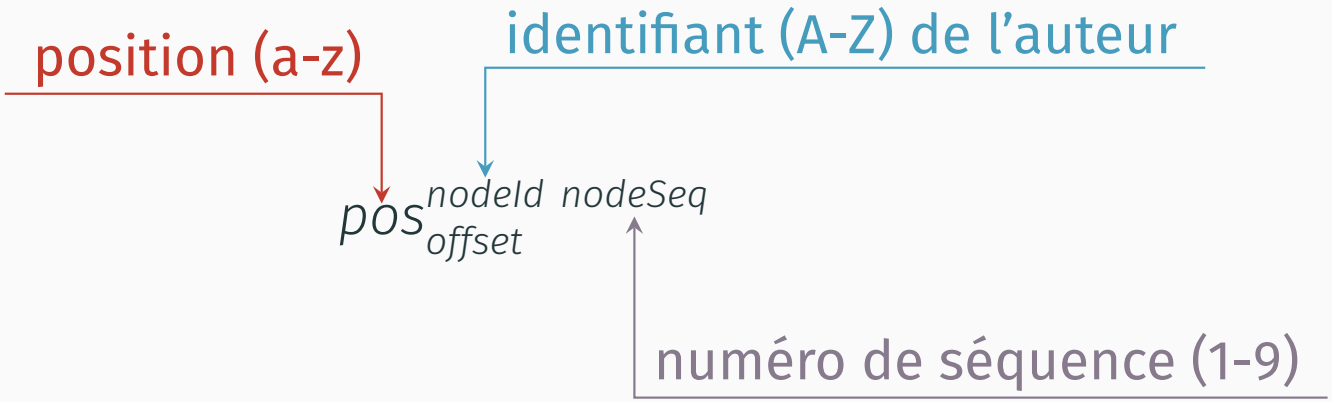
Exemples

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

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



Exemples

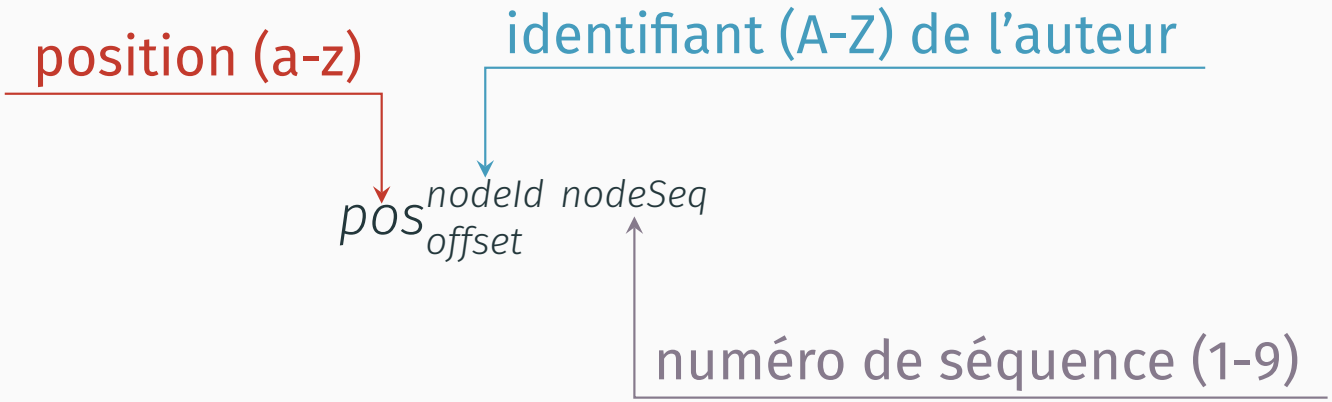
$$d_0^{F5} <_{id} m_0^{C1} <_{id} m_0^{C1} f_0^{E1}$$

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

Identifiant LogootSplit

Identifiant

- Composé d'un ou plusieurs tuples de la forme



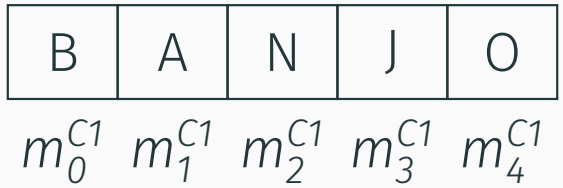
Exemples

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

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

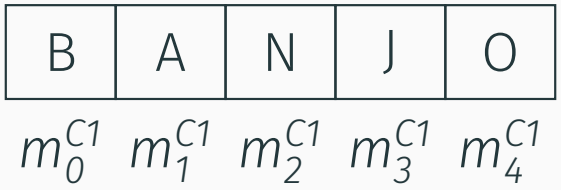
Bloc LogootSplit

- Coûteux de stocker les identifiants de chaque élément



Bloc LogootSplit

- Coûteux de stocker les identifiants de chaque élément



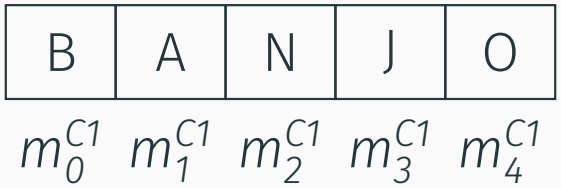
- Aggrège en un bloc éléments ayant identifiants contigus

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



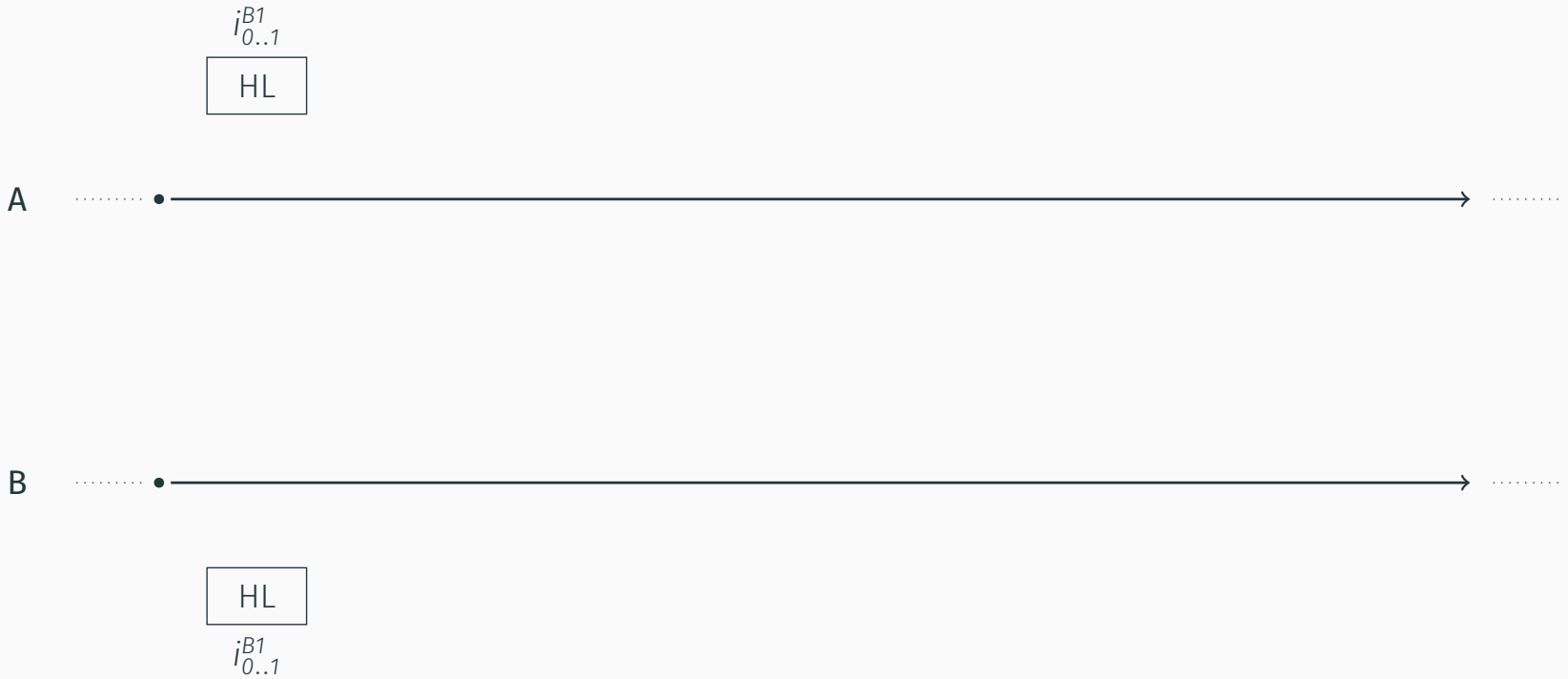
- Aggrège en un bloc éléments ayant identifiants contigus

Identifiants contigus

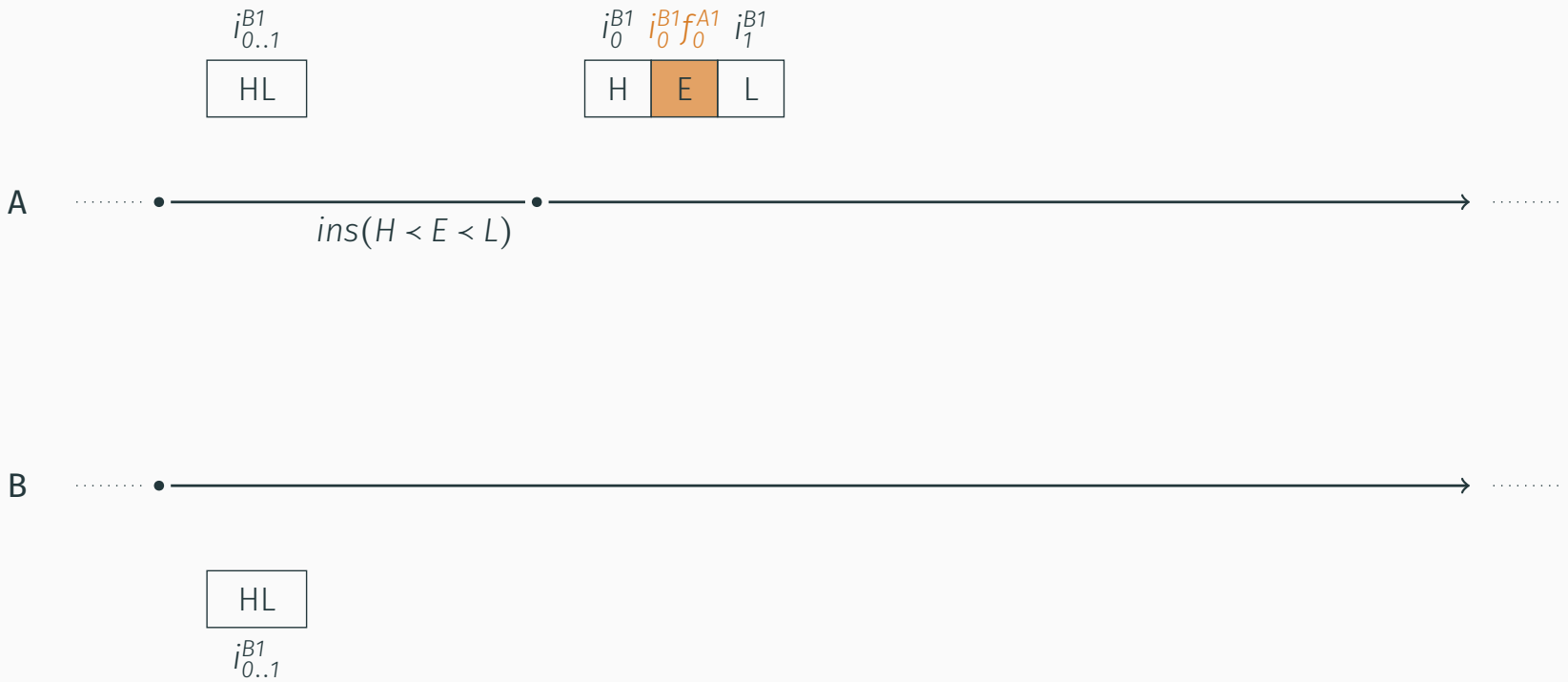
Deux identifiants sont contigus si et seulement si les deux identifiants sont identiques à l'exception de leur dernier offset et que leur derniers offsets sont consécutifs.

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

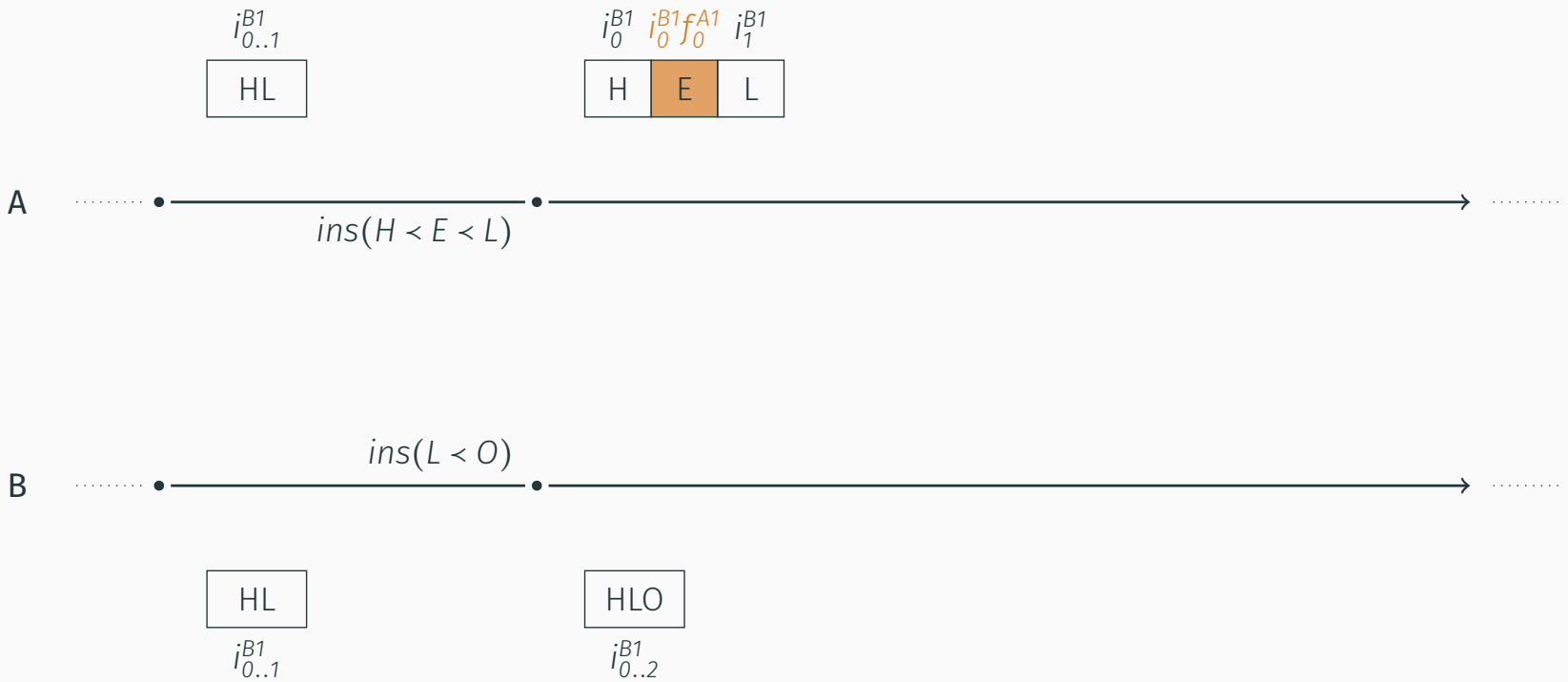
Exemple insertions concurrentes



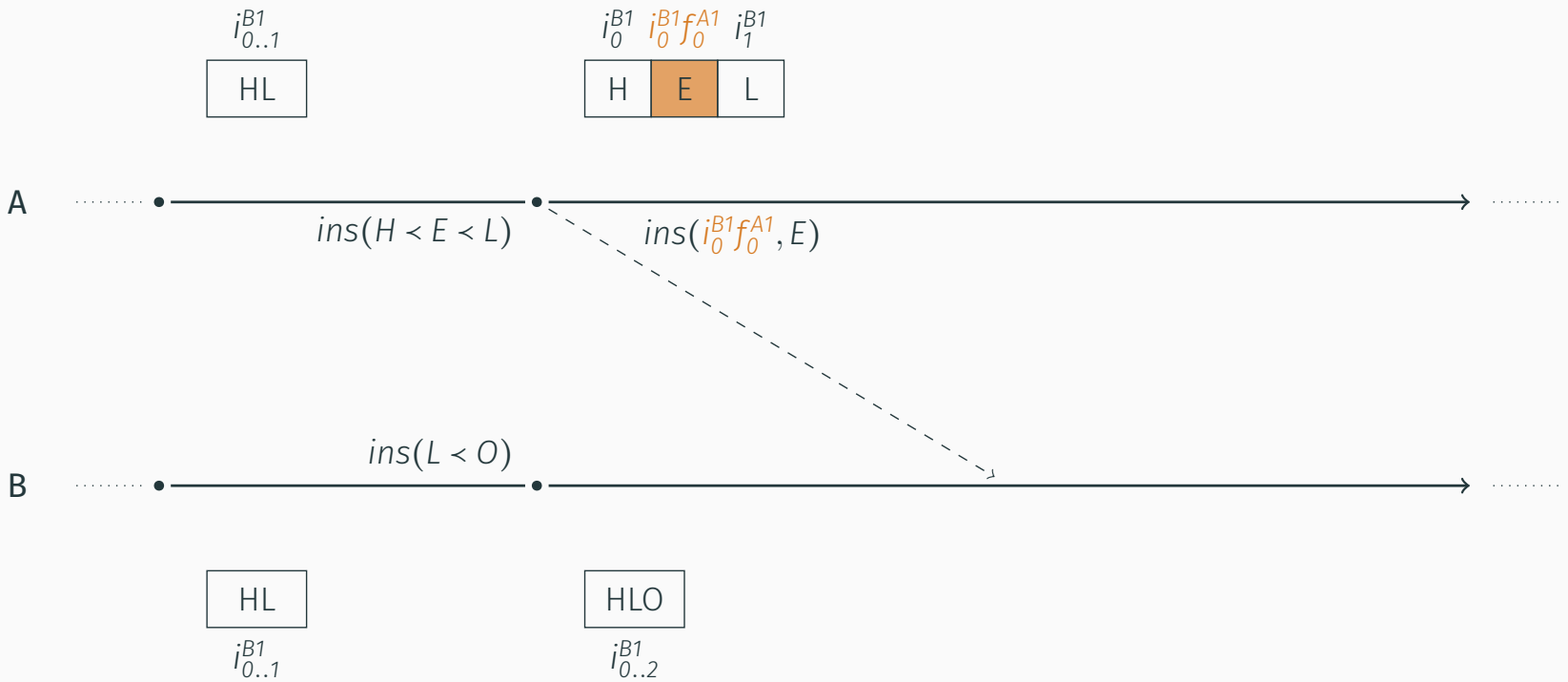
Exemple insertions concurrentes



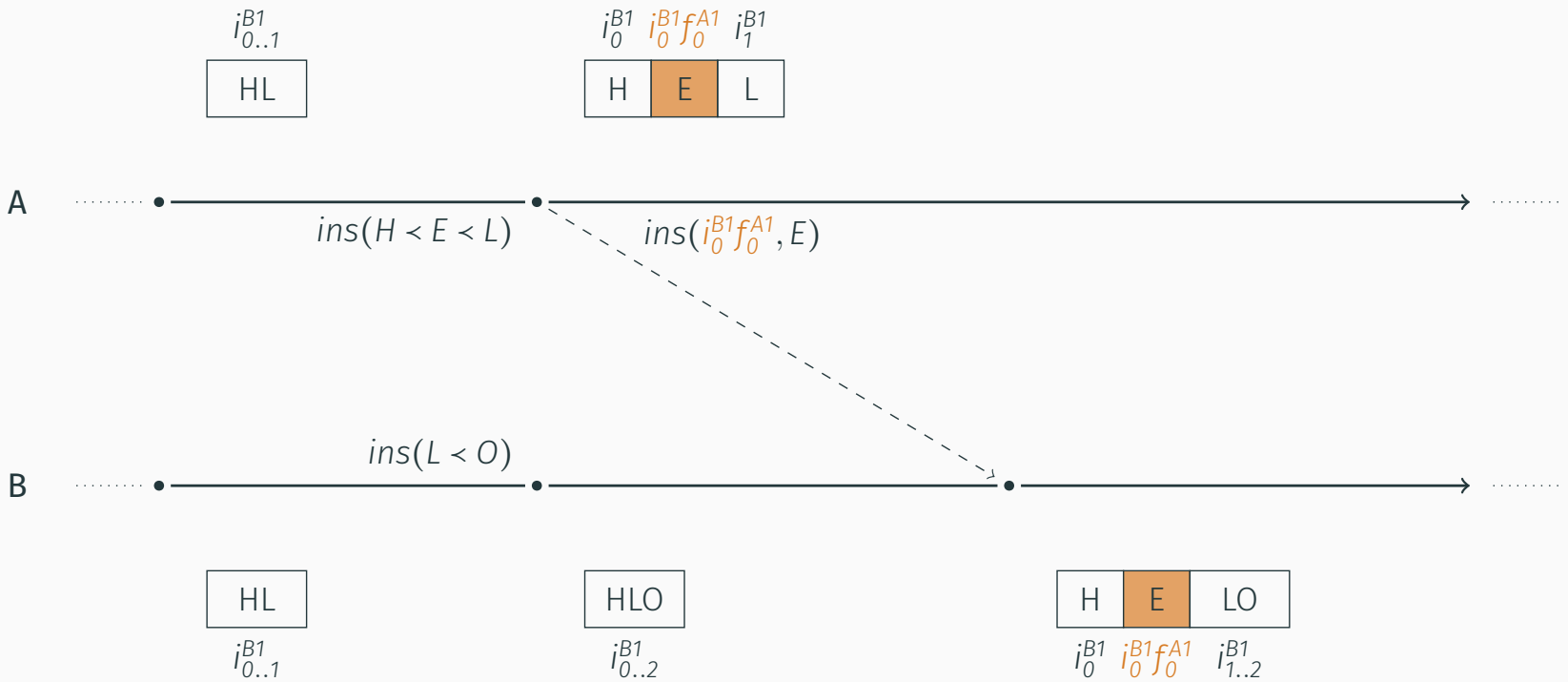
Exemple insertions concurrentes



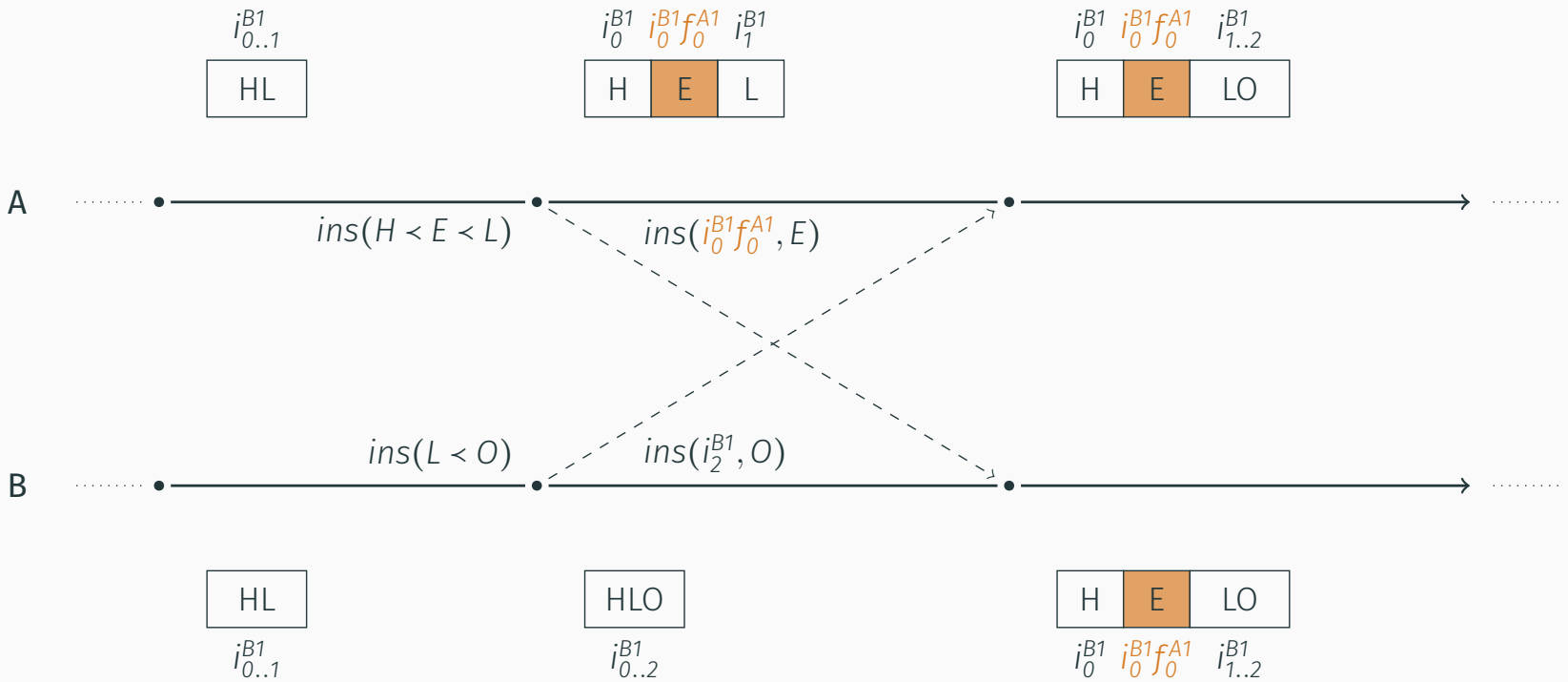
Exemple insertions concurrentes



Exemple insertions concurrentes



Exemple insertions concurrentes



Limites de LogootSplit

Sources de la croissance des métadonnées

- Augmentation non-bornée de la taille des identifiants
- Fragmentation de la séquence en un nombre croissant de blocs

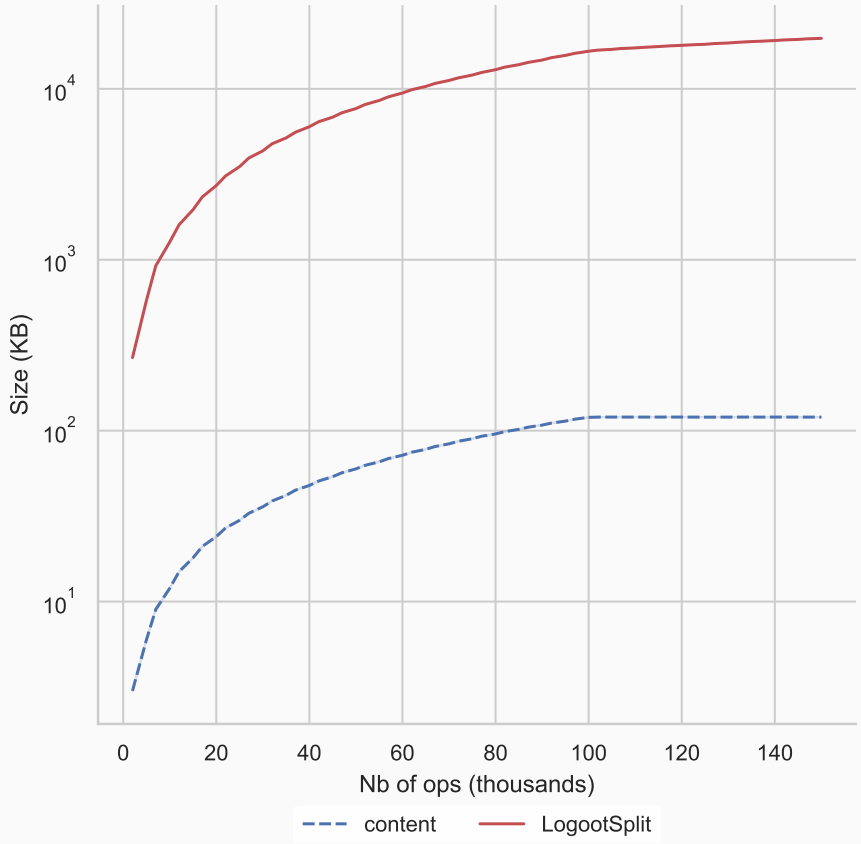
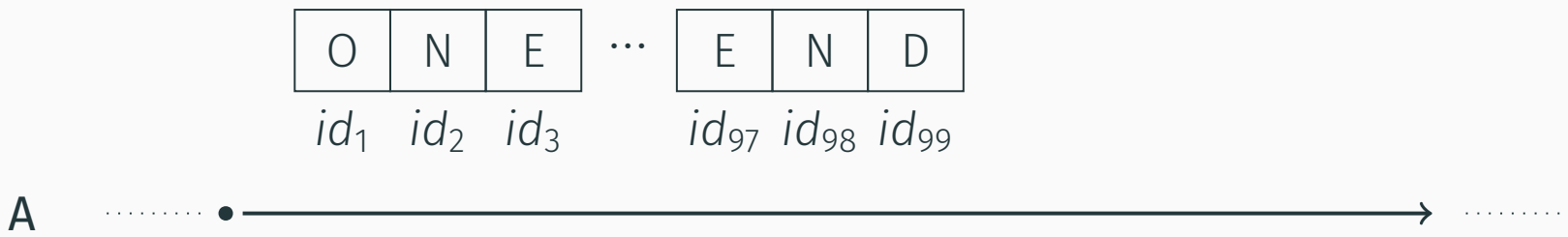


Figure 1 – Taille du contenu comparée à la taille de la séquence LogootSplit

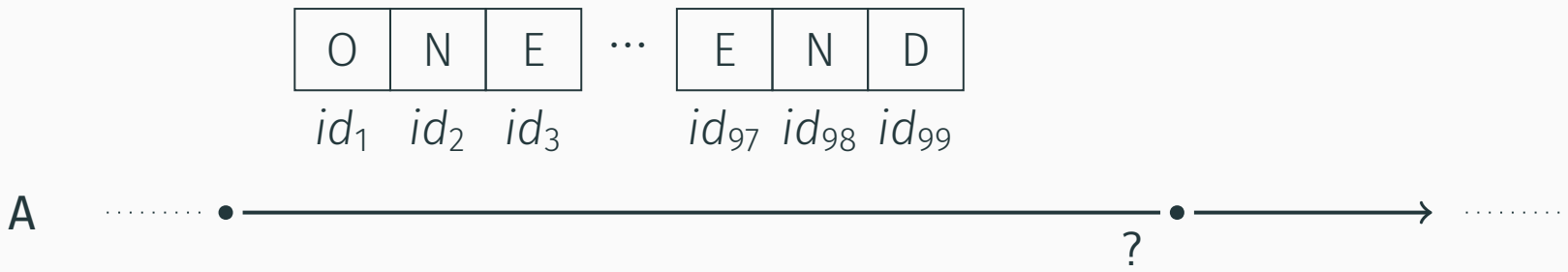
Diminution des performances du point de vue mémoire, calculs et bande-passante

Solution naïve



- Convertir l'état inefficient...

Solution naïve



- Convertir l'état inefficient...
- ...à l'aide d'une nouvelle opération ...

Solution naïve

- Convertir l'état inefficient...
- ...à l'aide d'une nouvelle opération ...
- ...en un état optimisé (identifiants de taille minimale, moins de blocs)

Mitigation du surcoût des CRDTs pour le type Séquence

L'approche core-nebula [5], pour Treedoc

- Ré-assigne des identifiants plus courts aux éléments
- Transforme les opérations *insert* et *remove* concurrentes…

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

Mitigation du surcoût des CRDTs pour le type Séquence

L'approche core-nebula [5], pour Treedoc

- Ré-assigne des identifiants plus courts aux éléments
- Transforme les opérations *insert* et *remove* concurrentes…
- …mais ne supporte pas opérations *rename* concurrentes
- Repose sur un algorithme de consensus pour décider du renommage

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

Mitigation du surcoût des CRDTs pour le type Séquence

L'approche core-nebula [5], pour Treedoc

- Ré-assigne des identifiants plus courts aux éléments
- Transforme les opérations *insert* et *remove* concurrentes…
- …mais ne supporte pas opérations *rename* concurrentes
- Repose sur un algorithme de consensus pour décider du renommage

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

Synthèse

Adaptation du mécanisme de renommage pour LogootSplit

Synthèse

Adaptation du mécanisme de renommage pour LogootSplit

- Opération *rename* permettant de minimiser le surcoût de l'état
- Mécanisme de détection des opérations concurrentes
- Algorithme pour intégrer l'effet d'une opération *rename* dans une opération *insert* ou *remove* concurrente

Synthèse

Adaptation du mécanisme de renommage pour LogootSplit

- Opération *rename* permettant de minimiser le surcoût de l'état
- Mécanisme de détection des opérations concurrentes
- Algorithme pour intégrer l'effet d'une opération *rename* dans une opération *insert* ou *remove* concurrente

Conception d'un mécanisme de résolution de conflits pour opérations *rename* concurrentes

Synthèse

Adaptation du mécanisme de renommage pour LogootSplit

- Opération *rename* permettant de minimiser le surcoût de l'état
- Mécanisme de détection des opérations concurrentes
- Algorithme pour intégrer l'effet d'une opération *rename* dans une opération *insert* ou *remove* concurrente

Conception d'un mécanisme de résolution de conflits pour opérations *rename* concurrentes

- Mécanisme pour désigner une époque comme l'époque cible, sans coordination
- Algorithme pour annuler l'effet d'une opération *rename*

Synthèse

Adaptation du mécanisme de renommage pour LogootSplit

- Opération *rename* permettant de minimiser le surcoût de l'état
- Mécanisme de détection des opérations concurrentes
- Algorithme pour intégrer l'effet d'une opération *rename* dans une opération *insert* ou *remove* concurrente

Conception d'un mécanisme de résolution de conflits pour opérations *rename* concurrentes

- Mécanisme pour désigner une époque comme l'époque cible, sans coordination
- Algorithme pour annuler l'effet d'une opération *rename*

Conception d'un mécanisme de suppression des époques obsolètes

Conclusion générale & Perspectives

Conclusion

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)

Conclusion

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

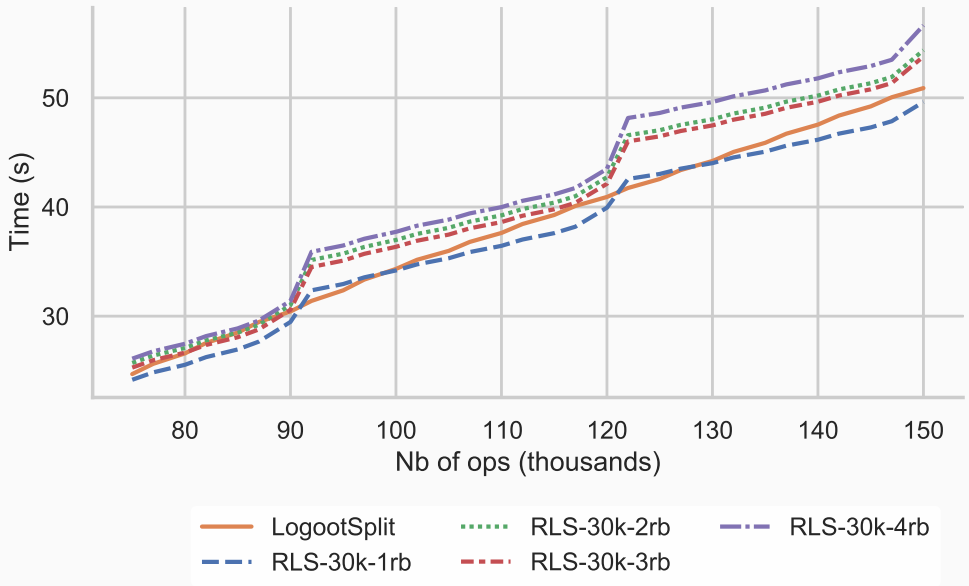
Limites de RenamableLogootSplit

- Surcoût de l'opération *rename*

Limites

Limites de RenamableLogootSplit

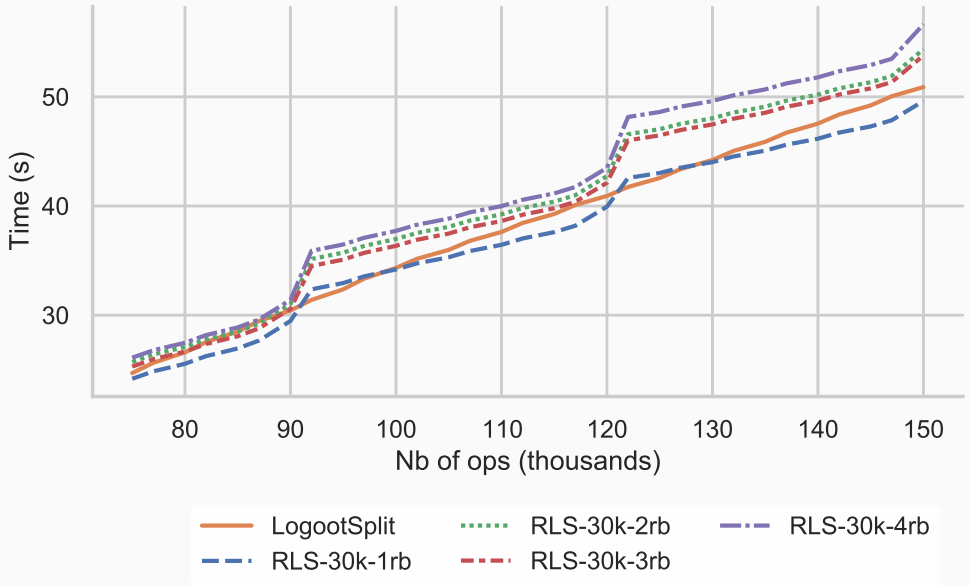
- Surcoût de l'opération *rename*



Limites

Limites de RenamableLogootSplit

- Surcoût de l'opération *rename*

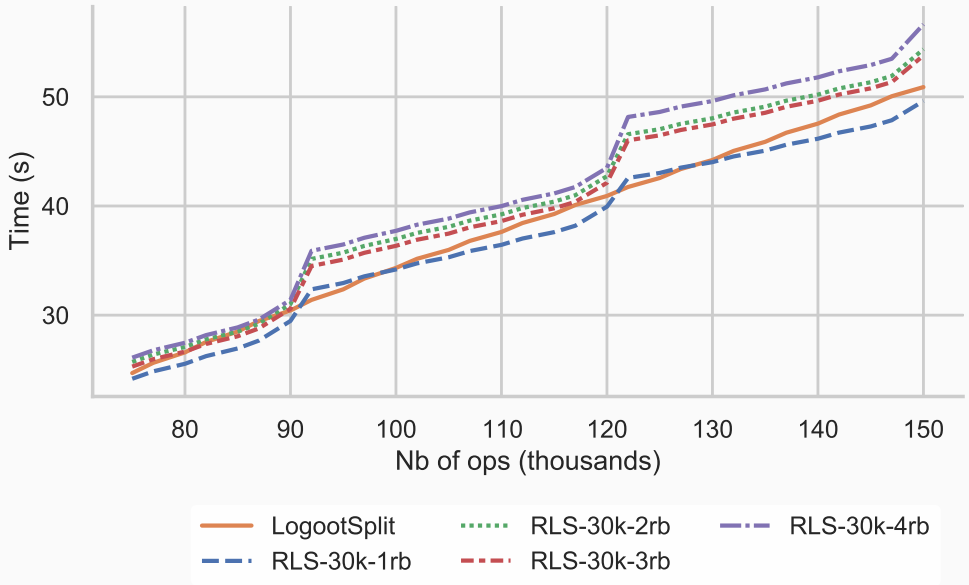


- Évaluations montrent que le temps d'intégration de l'opération *rename* peut atteindre 2s
- A privilégié la correction…
- …doit améliorer les performances (algorithme et implémentation)

Limites

Limites de RenamableLogootSplit

- Surcoût de l'opération *rename*



- Évaluations montrent que le temps d'intégration de l'opération *rename* peut atteindre 2s
- A privilégié la correction…
- …doit améliorer les performances (algorithme et implémentation)

- Stabilité causale requise pour supprimer les métadonnées

Perspectives

Perspectives autour de RenamableLogootSplit

- Comment définir des relations *priority* $<_\varepsilon$ qui minimisent les renommages vains ?
- Peut-on prouver formellement la correction RenamableLogootSplit ?

Perspectives

Perspectives autour de RenamableLogootSplit

- Comment définir des relations *priority* $<_\varepsilon$ qui minimisent les renommages vains ?
- Peut-on prouver formellement la correction RenamableLogootSplit ?

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 ?