Préservation d'intention dans les CRDT

Matthieu Perrin

École Normale Supérieure de Cachan – Antenne de Bretagne
Doctorant dans Aelos et GDD
directeurs :
Claude Jard
Achour Mostefaoui

Réunion interne, Aelos

16 mai 2013

Les objets séquentiels

Données

- états
- privées

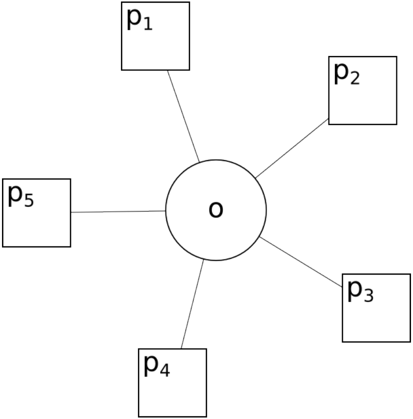
Méthodes

- opérations
- accessibles

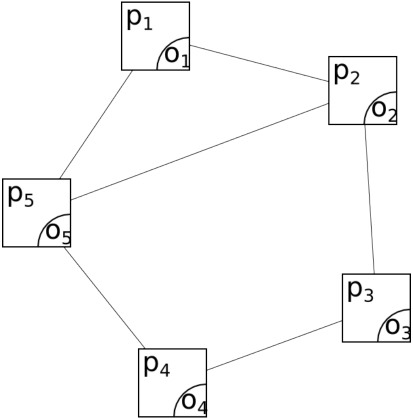
Spécification

- propriétés sur le comportement de l'objet
- à prouver

Les objets répartis



Les objets répartis



Présentation du système

n processus

- exécutions correctes
- asynchrones
- pannes franches

Moyens de communication

- passage de messages
- réseau fiable
- asynchrone

Le théorème CAP

Cohérence forte (Consistency)

Cohérence séquentielle :

- opérations totalement ordonnées
- ordre compatible avec l'ordre des processus

Disponibilité (Availability)

- les appels terminent

Résistence au Partitionnement

- des processus de partitions différentes ne peuvent pas communiquer

Le théorème CAP

Cohérence forte (Consistency)

Cohérence séquentielle :

- opérations totalement ordonnées
- ordre compatible avec l'ordre des processus

Disponibilité (Availability)

- les appels terminent

Résistence au Partitionnement

- des processus de partitions différentes ne peuvent pas communiquer

Impossible [Gilbert, Lynch, 2002]

Le théorème CAP

Cohérence forte (Consistency)

Cohérence séquentielle :

- opérations totalement ordonnées
- ordre compatible avec l'ordre des processus

Disponibilité (Availability)

- les appels terminent

Résistence au Partitionnement

- des processus de partitions différentes ne peuvent pas communiquer

Impossible [Gilbert, Lynch, 2002]

Cohérence inéluctable (eventual consistency)

Séparation des opérations :

- lectures
- écritures

Cohérence inéluctable (EC)

Si on arrête d'écrire, un jour, tous les processus liront les mêmes valeurs

Cohérence inéluctable forte (SEC)

Si deux processus ont reçu les mêmes mises à jour, ils sont dans le même état

Commutative Replicated Data Types

op-based object

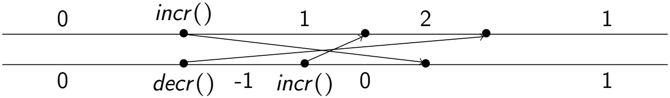
- $(S, s^0, q, t, u, (P))$
- S : ensemble d'états
- s^0 : état initial
- q : opération de lecture
- t, u : opérations d'écriture (local et distant)

CmRDT

$\forall s, \forall u, u', s \bullet u \bullet u' = s \bullet u' \bullet u$

Le compteur

payload : $i \in \mathbb{N}$;
initial : 0;
query *value*() : **return** i;
update *incr*() :
  atSource :
  downstream () : $i = i + 1$;
end
update *decr*() :
  atSource :
  downstream () : $i = i - 1$;
end



Convergent Replicated Data Types

state-based object

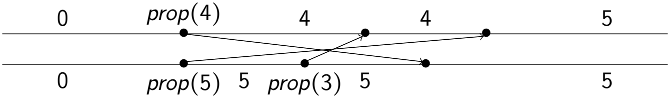
- $(S, \leq, s^0, q, u, m)$
- S : ensemble d'états
- $(S, \leq, m)$ est un demi-treillis
- s^0 : état initial
- q : opération de lecture
- u : opération d'écriture

CvRDT

$\forall s, \forall u, s \leq s \bullet u$

Optimisation

payload : $i \in \mathbb{N}$;
initial : 0;
query $\mathit{value}()$: **return** i;
update $\mathit{prop}(n \in \mathbb{N})$: $i = n$;
compare(i, j) : **return** $i \leq j$;
merge(i, j) : **return** $\mathit{max}(i, j)$;



Conflict-free Replicated Data Type

Théorème

Tout CmRDT peut être émulé par un CvRDT

Réciproque

Tout CvRDT peut être émulé par un CmRDT

Émulation de l'optimiseur

```
payload : i ∈ ℕ;
initial : 0;
query value() : return i;
update propose(n ∈ ℕ) :
    atSource :
    downstream (n) :
        if n > i then
            i = n
    end
end
```

Émulation du compteur

payload : $P[n] \in \mathbb{N}, N[n] \in \mathbb{N}$;
initial : $[0, ..., 0], [0, ..., 0]$;
query *value*() : **return** $\sum_{i=0}^{n-1} P[i] - N[i]$;
update *incr*() : $P[k] = P[k] + 1$;
update *decr*() : $N[k] = N[k] + 1$;
compare$((P, N), (P', N'))$:
  return $\forall i, P[i] \leq P'[i] \wedge N[i] \leq N'[i]$;
end
merge$((P, N), (P', N'))$:
  return $[max(P[i], P'[i])], [max(N[i], N'[i])]$;
end

L'ensemble partagé

Définition

Un état : $S \in \mathcal{P}(E)$

Deux opérations :

- insert(x)
- remove(x)

CmRDT

- insert(x); remove(x) $\Rightarrow x \notin S$
- remove(x); insert(x) $\Rightarrow x \in S$

CvRDT

$$\{S = \varnothing\} insert(x); \{S = \{x\}\} remove(x); \{S = \varnothing\}$$

2P-Set

payload : $S \subset E \cup E^{\dagger}$;
initial : $\varnothing$;
query *value*() : **return** $\{x \in E | x \in S \wedge x^{\dagger} \notin S\}$;
update *insert*$(x \in E)$:
  atSource :
  downstream (x) : $S = S \cup \{x\}$;
end
update *remove*$(x \in E)$:
  atSource :
  downstream (x) : $S = S \cup \{x^{\dagger}\}$;
end

OR-Set

payload : $S \subset (E \times \mathbb{N}) \cup (E \times \mathbb{N})^\dagger$;
initial : $\varnothing$;
query *value*() : **return** $\{x \in E | \exists n : (x,n) \in S \wedge (x,n)^\dagger \notin S\}$;
update *insert*$(x \in E)$:
  atSource : $n = unique()$;
  downstream (x,n) : $S = S \cup \{(x,n)\}$;
end
update *remove*$(x \in E)$:
  atSource : $R = \{n \in \mathbb{N} | (x,n) \in S\}$
  downstream (x,R) : $S = S \cup (\{x\} \times R)^\dagger$;
end

OR-Set – Version avec causalité

payload : $S \subset (E \times \mathbb{N})$;
initial : $\varnothing$;
query *value*() : **return** $\{x \in E | \exists n : (x, n) \in S\}$;
update *insert*$(x \in E)$:
  atSource : $n = unique()$;
  downstream (x, n) : $S = S \cup \{(x, n)\}$;
end
update *remove*$(x \in E)$:
  atSource : $R = \{n \in \mathbb{N} | (x, n) \in S\}$
  downstream (R) : $S = S \setminus (\{x\} \times R)$;
end

Précondition : réception causale

Rôle de la causalité

CvRDT

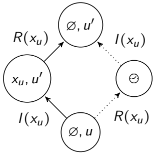
- Supprime des états dans le treillis

Garde-t-on un treillis ?

CmRDT

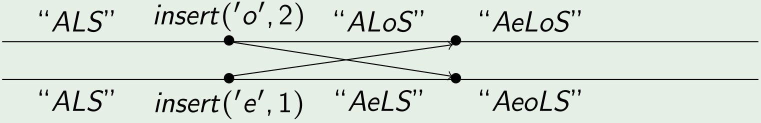
- Empêche certains ordres entre les opérations

Quelles commutations doit-on conserver ?



L'intention de Sun

Exemple : l'édition collaborative

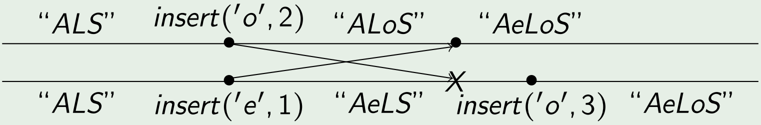


Définition

L'intention est l'effet d'une opération sur l'état de celui qui la soumet.

L'intention de Sun

Exemple : l'édition collaborative



Définition

L'intention est l'effet d'une opération sur l'état de celui qui la soumet.

Principe des permutations équivalentes

Conformité à la spécification séquentielle

$$\{P\}u; u'\{Q\} \wedge \{P\}u'; u\{Q'\} \wedge Q \Leftrightarrow Q' \Rightarrow \{P\}u||u'\{Q\}.$$

L'ensemble partagé

- $I(x)||I(y)$: ✓
- $R(x)||R(y)$: ✓
- $I(x)||R(y)$: ✓
- $I(x)||R(x)$: X
  - insert wins
  - remove wins
  - error state
  - ...

Limites du principe des permutations équivalentes

- Trois états : A, B, C
- Trois opérations : a, b, c
  - $\{true\}a\{A\}$
  - $\{true\}b\{B\}$
  - $\{true\}c\{C\}$
- Elles ne commutent pas :
  - $\{true\}b||c\{A\}$
  - $\{true\}a||c\{B\}$
  - $\{true\}a||b\{C\}$

Limites du principe des permutations équivalentes

- Trois états : A, B, C
- Trois opérations : a, b, c
  - $\{true\}a\{A\}$
  - $\{true\}b\{B\}$
  - $\{true\}c\{C\}$
- Elles ne commutent pas :
  - $\{true\}b||c\{A\}$
  - $\{true\}a||c\{B\}$
  - $\{true\}a||b\{C\}$

Exécution de $a||b||c$?

Limites du principe des permutations équivalentes

payload : $i \in \mathbb{N}$;
initial : 0;
query *value*() : **return** i;
update *incr*() : $i = i + 1$;
compare(i, j) :
  return $i \leq j$;
end
merge(i, j) :
  return $max(i, j)$;
end

$$\{n\} incr; incr \{n+2\}$$

$$\{n\} incr \| incr \{n+1\}$$

Limites du principe des permutations équivalentes

Cette définition de l'ensemble

$$\{true\}I(x)\{x \in S\}$$

$$\{true\}R(x)\{x \notin S\}$$

$$\{true\}I(x)\|R(x)\{S = \emptyset\}$$

Est-elle conforme à sa spécification séquentielle ?

De quoi dépend le comportement d'un objet ?

Exécution

Ensemble de lectures et d'écritures partiellement ordonnées selon l'ordre causal.

Ce que l'on veut spécifier

Résultat des lectures

$$\mathcal{O} = (W, \leq) \to Q$$

Quelles fonctions correspondent aux CRDT ?

L'objet causal

payload : $ops \subset \mathbb{N}^2$, $order \subset \mathbb{N}^4$;
initial : $\emptyset, \emptyset$;
query $q_i()$: **return** $f_i(ops, order)$;
update $u_i()$:
  $unique = \mathcal{U}$;
  $order = order \cup \{(i, unique)\} \times ops$;
  $ops = ops \cup \{(i, unique)\}$;
end
compare$((op, or), (op', or'))$:
  return $op \subset op' \wedge or \subset or'$;
end
merge$((op, or), (op', or'))$:
  return $(op \cup op', or \cup or')$;
end

Définition de fonctions

- exécution = graphe orienté acyclique
- exploration de graphe : largeur / profondeur
- modèle intention–résolution

  Intention : action d'une opération sur un état
  Résolution : état sur lequel une opération est exécutée

$\{s_0\}o_1; (o_2 || o_3); o_4$

- $s_0 \xrightarrow{\mathcal{I}(o_1)} s_1$
- $s_1 \xrightarrow{\mathcal{I}(o_2)} s_2$
- $s_1 \xrightarrow{\mathcal{I}(o_3)} s_3$
- $\mathcal{R}(s_3, s_4) \xrightarrow{\mathcal{I}(o_4)} s_4$

L'ensemble partagé

Spécification

- état abstrait : $S \subset E$
- état initial : $\varnothing$
- intention de insert : $S := S \cup \{x\}$
- intention de remove : $S := S \setminus \{x\}$
- résolution$(S_1, ..., S_n)$: $S := \bigcup_{i=1}^{n} S_i$

Preuve pour OR-set

Soit $(W, \leq)$ une exécution,

- dans la spécification : $x \in S \Leftrightarrow \exists I_x \in W, \nexists R_x \in W, I_x \leq R_x$
- dans OR-set : $x \in S.value() \Leftrightarrow \exists I_x \in W, \nexists R_x \in W, I_x \leq R_x$

Conclusion

L'intention

- les approches actuelles sont limitées
- les opérations ne peuvent pas être spécifiées individuellement

Travaux en cours

- spécification avec des ordres partiels
- preuves encore compliquées

Avoir une spécification séquentielle

Nouvelle définition

Un objet a une spécification séquentielle si ses écritures sont sérialisables

Lien avec le principe des permutations équivalentes

- spécification séquentielle $\Rightarrow$ PPE
- réciproque fausse

Peut-on s'en servir pour simplifier la spécification ?

Gérer la cohérence

Objets séquentiels

- fonction d'une séquence dans un état
- cas particulier
- peut-on les obtenir à partir d'un CRDT ?

Calculabilité des systèmes répartis

- système avec consensus $\equiv$ machine de Turing
- système avec majorité $\equiv$ machine de Turing
- système à sémantique faible $\equiv$ machine de Turing
- sémantique faible $<$ majorité $<$ consensus