ÉCRITURE ET EXÉCUTION RÉPARTIE DE SCÉNARIOS INTERACTIFS

Auteur 1
Organisme
Adresse électronique

Auteur 2
Organisme
Adresse électronique

Auteur 3
Organisme
Adresse électronique

Résumé

Le résumé doit être placé en haut de la colonne gauche et doit contenir entre 150 et 200 mots.

1. INTRODUCTION

1.1. Problématique

1.2. Études de cas

-> installation en son réparti ?

- Problème de la latence entre i-score et raspberry, + utilisation de bande passante.

- Cas des applis de téléphone : un objet qui s'exécute sur plusieurs machines en parallèle dont on veut aggréger les résultats

- Donner la possibilité de faire un dialogue (boucle avec un coup un trigger qui vient de A, un coup un trigger qui vient de B, etc)

- Approche alternative avec application mobile.

- Faciliter l'écriture de programmes qui s'exécutent sur des machines embarquées

- Lister les cas. D'abord non interactif (partage uniquement des processus), puis interactif.

1.3. État de l'art

1.3.1. Rappel du modèle d'i-score

Vocabulaire. Prendre article SMC.

1.3.2. Répartition à l'exécution

1.3.3. Horloges

Problème de l'horloge :

— Si on utilise l'horloge système, pas de garantie qu'elle soit bien synchronisée. Et NTP pas dispo partout (on n'a pas forcément les droits sur un téléphone pour changer l'heure). Horloge i-score se resynchronise déjà en cas de délai sur une machine.

— Si on utilise une horloge interne, problème de la synchronisation avec l'horloge système (pour le tick)

http://queue.acm.org/detail.cfm?id=2745385 http://radar.oreilly.com/2012/10/google-spanner-relational-database.html http://www.ntp.org/ntpfaq/NTP-s-sw-clocks-quality.htm

Faire la relation entre notre système et le problème CAP : https://en.wikipedia.org/wiki/PACELC_theorem

Horloges logiques : équivalent à envoyer une timestamp à chaque tick (on arrive donc à un méchanisme de vector clock approximatif).

Synchro d'horloges de deux machines dont une qui ralenti, si elles exécutent le même scénario ? Il faut les recaler régulièrement.

Possibilité : externe (NTP, etc) ou interne.

Recaler dès qu'on accumule du retard ? On peut introduire un recalage sur la master clock entre chaque tick.

1.3.4. Problématique de la sécurité

1.3.5. Synchronisation de médias

- pas pour l'instant... doivent être au même endroit sur la même machine. Ableton Link ? Netjack ?

2. APPROCHE

2.1. Notions introduites

2.1.1. Groupe

2.1.2. Client

2.2. Topologie

Maître-esclave pour l'édition.

Pour l'exécution : dans certains cas deux machines peuvent échanger directement. Il faut décrire ces cas.

Si groupes avec plusieurs machines : "group leader" ? responsable de la communication ? avoir un "plus court chemin" permanent ?

2.3. Fondements

Première approche basées sur réseau de Petri.

Puis migration du modèle d'i-score maintenant autonome.

Il est possible d'implément ce méchanisme de répartition purement avec les primitives dont on dispose dans i-score, mais c'est prohibitoire pour les utilisateurs.

Différentes possibilités de répartition sont disponibles selon les types de processus.

2.4. Modèle

i-score se base sur une approche orientée document.

On choisit de répartir nos objets autour d'un même document partagé par toutes les instances du réseau, à la manière des logiciels d'édition tels que Google Docs.

Cela ne pose pas de problème de mémoire, même sur de l'embarqué : les scénarios i-score sont très compacts (en général quelques dizaines de kilo-octets).

2.4.1. Répartition de l'édition du modèle

- Partage de file undo - redo. Transformations opérationnelles.
- Autres approches : partage du modèle lui-même.

3. DESCRIPTION DE L'EXÉCUTION

3.1. Processus de contenu

Automation, mapping, JS, audio...

Le processus s'exécute tel quel pour toutes les machines auquel il est assigné, et ne s'exécute pas pour les autres.

3.2. Processus Scenario

3.2.1. Cas interactif

Trois niveaux de partage :

— Aucun partage : les machines n'étant pas associées à ce processus ne l'exécutent pas, celles qui y sont associées l'exécutent toutes de manière indépendante. Note : scénarios hiérarchiques ?

— Partage partiel : il peut y avoir plusieurs lignes temporelles dans un même scénario qui doivent se resynchroniser. Les annotations donnent l'emplacement d'exécution, et de vérification des expressions.

— Partage complet : il n'y a qu'une seule "ligne temporelle" pour toutes les machines. Les annotations de machines indiquent l'emplacement d'exécution.

Deux niveaux de synchronisation :

— Asynchrone : dès qu'une information est disponible dans le système (par exemple « une expression se vérifie »), elle est propagée le plus vite possible aux autres noeuds qui doivent appliquer le résultat de cette information.

— Synchrone : lorsqu'une information est disponible dans le système, elle est propagée de manière à ce que la date absolue de réalisation soit la même pour toutes les machines.

Problème transverse : ignorer ou non la hiérarchie ?

Différence entre synchro partielle et complète :

— Exécution des triggers

— Validité des conditions

— Changement de vitesse d'exécution des contraintes temporelles

Dans tous les cas, cela se fait au niveau des groupes, et il peut y avoir plusieurs machines par groupe. Donc il faut un méchanisme de décision au niveau du groupe. -> pourquoi ne pas avoir la même chose au niveau des mappings (et pour tout ce qui n'est pas fixé de manière générale ?)

3.2.2. Cas non-interactif

On peut soit utiliser les méthodes de synchronisation décrites précédemment, mais une possibilité s'ajoute :

Comme on connait les dates effectives auxquelles les objets sont sensés s'exécuter, on peut les fixer à l'avance sur chaque machine.

— Avantage : s'il y a une déconnection, cela va continuer à marcher.

— Inconvénient : il suffit d'un peu de délai pour que le début de B peut survenir avant la fin de A. Il est particulièrement important dans ce cas de garder les horloges synchronisées.

3.2.3. Expressions et interactivité

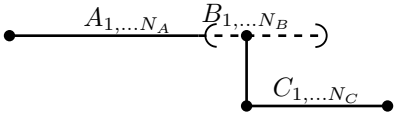
Dans le cas ou un scénario est exécuté intégralement en parallèle par différentes machines, il n'y a pas de problème : chacune vérifie les expressions en fonction des données dont elle dispose, et les valide à l'instant ou elle le souhaite. C'est utile si par exemple on veut avoir plusieurs téléphones qui font tous tourner un scénario semblable, mais chaque individu peut choisir de faire avancer le scénario au rythme ou il le souhaite.

Dans le cas ou on a un partage de certaines lignes temporelles par certaines machines, le problème de la synchronisation des expressions se pose.

— Prise de décision : s'il n'y a qu'une seule ligne temporelle, chaque expression doit n'avoir qu'une valeur de vérité. Il doit donc y avoir un consensus sur la valeur de cette expression. De même pour la vitesse d'exécution des éléments.
  Plusieurs algorithmes de consensus sont possibles :
  — Au moins une machine valide une expression. Et spécifiquement, dans le cas des points d'interaction : la première machine qui valide une expression donne le résultat.
  — Toutes les machines valident une expression.

— On peut supposer que des sous-parties d'un scénario pourraient être exécutées entièrement par une machine ou groupe.

Ordonnancement : par timestamps.



Scénario 1 : Des groupes A et B exécutent les deux contraintes temporelles et un groupe C l'expression

Le scénario 1 peut se résoudre de plusieurs manières.

— Si l'exécution du noeud est résolue de manière asynchrone, toutes les machines de C envoient l'information de trigger à toutes les machines de A et B, sans garantie d'ordre. Les triggers se déclenchent immédiatement.
Note : si on veut garantir un ordre, ne peut-on pas faire : C stoppe A qui démarre B ?
— Si l'exécution du noeud est résolue de manière synchrone,

3.2.4. Consensus

Paxos, Raft sont des algorithmes permettant de garantir un consensus.

S'il y a synchronisation, on sépare la synchronisation du consensus, de la synchronisation de l'exécution qui suit la résolution de ce consensus :

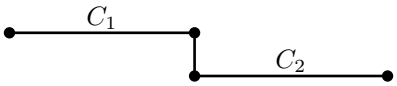
1. Les nœuds impliqués dans l'expression décident de la valeur de vérité.
2. Une fois cette valeur connue, tous les nœuds précédant, suivant, et impliquant l'expression sont inclus dans la décision de la date d'exécution.

Note : abstraire la notion de consensus au sein d'un groupe.

Cas avancé et non traité ici : si un scénario est exécuté en parallèle par toutes les machines, chaque groupe peut avoir un consensus différent sur chaque trigger. On peut rajouter un niveau d'indirection. Le plus général : chaque objet peut être exécuté par N groupes pouvant posséder chacun N clients. Si un client est dans deux groupes, comment gérer le conflit ? bof.

3.3. Introduction de primitives réparties dans i-score

3.3.1. Répartition des contenus dans le cas non-interactif



Scénario 2 : Deux contraintes temporelles se suivent

— Cas partagé : toutes les machines vont exécuter l'ensemble du scénario en parallèle.
— Cas synchronisé : chaque machine va exécuter une partie du scénario : par exemple, une machine exécute C_1 et une autre C_2.

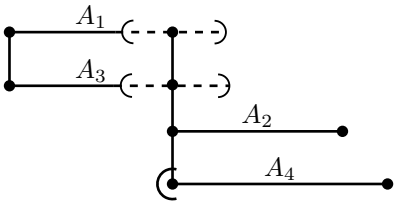
On sépare donc les objets structurants, des objets possédant des données (par exemple, une automation).

3.3.2. Répartition des contenus dans le cas interactif

Un problème de synchronisation

- On souhaite modifier le moins possible le modèle pour les utilisateurs

- Notion de groupe. Intérêt d'un groupe : s'abstrait des clients qui peuvent se déconnecter / reconnecter.



Scénario 3 : Interactions et conditions

- Problème des devices. Note : si on se permet d'envoyer des messages OSC quelconques (i.e. pas dans l'arbre), ça peut simplifier des choses. Aussi, permettre devices qui sont juste en mode envoi ? Il faut un autre proto que OSC ou Minuit...

- implémentation : on rajoute des délais dans les expressions.

Cas synchrone : * Sous-expression devient vraie * Expression fixe une date possible pour le réseau et notifie les autres puis se fixe au bon délai.

Cas asynchrone : * Sous-expression devient vraie * Expression envoie un message aux autres et se lance immédiatement. Il faut définir sur quelle machine l'expression est validée.

- Cas de la borne max : trigger, bof

Groupes pour triggers : problème du consensus. Voir notamment Paxos, Raft...

Pour l'instant : group leader ? ou bien tous se communiquent l'information et prennent la décision en fonction de cette information ?

- Latence : prendre moyenne et écart-type sur les dix dernières valeurs ? Ou juste dernière valeur ? En LAN gigabit on est en général $<$ à une milliseconde.

- Pour l'édition temps-réel, on doit permettre d'appliquer des "filtres" (par exemple qui vont rajouter une sur-expression, etc)

3.4. Méchanismes de synchronisation

- Cas possibles :

C1 ——t—— C3
C2 ———|——— C4

* Mode free ou synchronisé : pour chaque client ?

Cas 1 :

C1 ——t—— C3

C1 libre, C3 libre : dès que t est vrai, un message est envoyé au master qui envoie à C3 le trigger. C1 s'est déjà arrêté. Voire, ils peuvent se trig eux même en asynchrone.

C1 synchro, C3 synchro : dès que t est vrai, C1 calcule la date minimale à laquelle C3 peut être notifié, envoie le message et fixe ce temps de son côté.

Cas 2 :

C1 ——t—— C3
C2 ———|

* Synchronisation de la fin
* Synchronisation du début de la suite

En pratique, on n'implémente que la possibilité de synchro / désynchro toute une time node ; dire qu'une granularité plus fine est possible mais que l'intérêt n'apparaît pas en pratique dans les applications (et complexifierait l'UI pour rien).

Question principale : pour un time-node, comment choisit-on sur quelle machine une condition doit être vérifiée ? Possibilité de conditions groupées : faut-il que ce soit interne au formalisme (i.e. une case à cocher) ou externe (on dit explicitement machine1:/truc && machine2:/truc)

3.4.1. Cas ou il y a plusieurs machines dans un groupe

Attributs d'une expression :

— Qui vérifie cette expression : chaque machine individuellement, toutes les machines d'un groupe, n'importe quelle machine d'un groupe

Séparer les expressions et les time-node : notamment pour le cas de la borne max, ou on doit avertir d'autres machines par la suite.

Deux axes :

Décision \ Sync	Oui	Non
Locale	1	2
Partagée	3	4

— Décision locale : Pour un trigger, soit tout le scénario est local (donc même contrainte avant / après et certains ordis n'exécutent pas du tout ce scénario), soit le premier à avoir un résultat avertit les suivants.
— Décision partagée : tous les éléments d'un groupe doivent prendre une décision avec une politique donnée : soit tous doivent vérifier la condition, soit un seul.
— Décision synchrone : On veut que les éléments s'arrêtent et démarrent le plus proche possible d'une même date en wall clock.
— Décision asynchrone : On veut que les éléments s'arrêtent et démarrent le plus vite possible dès qu'une information est disponible.

1. Décision locale synchrone : pas de sens
2. Décision locale asynchrone : pas de sens...
3. Décision partagée synchrone : toutes les machines attendent qu'un unique choix soit effectué pour le résultat de l'expression
4. Décision partagée asynchrone

Possibilité : synchronisation via démon externe (PTP, NTP...), mais pas toujours possible (on ne peut pas supposer que l'utilisateur a les droits pour changer l'horloge sur sa machine).

Synchronous ethernet

Ableton Link : synchro sur les ticks musicaux

Avoir une horloge propre à i-score ? Mais du coup maintenant il faut la synchroniser à l'horloge système.

Ce qu'on fait : ping régulier vers chaque client (toutes les 100 millisecondes)

Quand quelque chose doit se synchroniser, on dit à chaque machine à quel instant il est supposé arriver par rapport à son horloge système.

Extension via système de composants

Quand un client reçoit un ordre pour un timenode à t, il l'applique dès que t <= local(t) (modulo un tick ?)

On ne synchronise qu'à chaque point d'entrée / de sortie d'un groupe de contraintes / time nodes

3.5. Extension des possibilités d'écriture

* Paramètres partagés
* Pattern matching sur addresses

4. PERFORMANCE

- Comparaison de l'algorithme "simple" et de l'algorithme avec retard

5. CITATIONS

Toutes les références bibliographiques des citations devront être listées dans la section "References", numérotées et en ordre alphabétique. Toutes les références listées devront être citées dans le texte. Quand vous vous référez au document dans le texte, précisez son numéro [?].