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. Contexte

1.1.1. Horloges

1.1.2. Synchronisation de médias

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

1.2. État de l'art

1.2.1. Rappel du modèle d'i-score

Vocabulaire. Prendre article SMC.

1.2.2. Répartition à l'édition

1.2.3. Répartition à l'exécution

1.2.4. Lien entre la répartition à l'édition et la répartition à l'exécution

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

3. APPROCHE

- Groupe, client
- Maître-esclave
- Dire qu'on peut toujours le faire "à la main" dans i-score, mais que c'est long et complexe.
- La manière dont la répartition se fait dépend du processus.

3.1. Édition

- Partage de file undo - redo
- Un seul "document" logique sur lequel tout le monde travail. Comme google docs.

3.2. Exécution

3.2.1. 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.

3.2.2. Processus Scenario

Trois niveaux de synchronisation :
— Aucune synchronisation : 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 ?
— Synchronisation partielle : 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.
— Synchronisation complète : il n'y a qu'une seule "ligne temporelle" pour toutes les machines. Les annotations de machines indiquent l'emplacement d'exécution.

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 ?)

3.2.3. 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 éléments de 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, de une, plusieurs, ou toutes machines, sur la valeur de cette expression. De même pour la vitesse d'exécution des éléments.
— On peut supposer que des sous-parties d'un scénario pourraient être exécutées entièrement par une machine ou groupe.

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

3.2.4. Horloges

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.

3.3. Introduction de primitives réparties dans i-score

3.3.1. Répartition des contenus dans le cas non-interactif

— 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
- Problème des devices
- 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 && machin

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 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

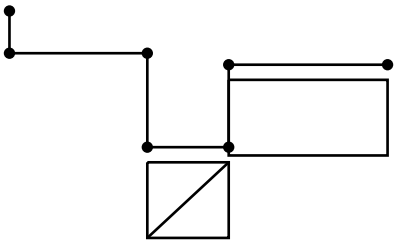
5. CITATIONS

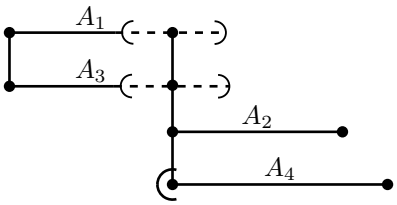
Toutes les références bibliographiques des citations devront être listées dans la section "References", numérotées et en ordre alphabétique. Toutes les références listées devront être citées dans le texte. Quand vous vous référez au document dans le texte, précisez son numéro [1].

6. REFERENCES

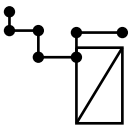
[1] Author, E. "Titre du papier", Proceedings of the International Symposium on Music Information Retrieval, Plymouth, USA, 2000.

[2] Untel, A. Titre du livre. L'Armada, Paris, 2005.

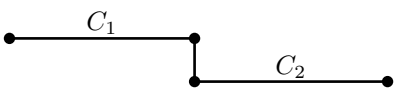




Scénario 2 : Interactions et conditions



Décision \ Sync	Oui	Non
Locale	1	2
Partagée	3	4



Scénario 1 : Deux contraintes temporelles se suivent