Fichiers à l'Unison
Données en HARMONY

Logiciels de Base, Episode III:
Les Champs de la Réconciliation

Alan Schmitt

INRIA Rhône-Alpes

18 mai 2005

Un scénario classique . . .

- Travail collaboratif sur un projet
  - plusieurs documents, rangés dans plusieurs dossiers
- Copie du projet sur ordinateur portable
- Dans le train, une personne modifie des documents
- Pendant ce temps, d'autres personnes travaillent sur le projet
- Au retour, comment réconcilier les deux versions ?

Réplication et Synchronisation

- Réplication de données
  - pour un accès plus rapide
  - pour un accès plus pratique (son environnement)
  - pour un accès en mode déconnecté (train, avion)
  - pour garder une copie (sauvegarde)
- Ces replicas peuvent être modifiés indépendemment
- $\Longrightarrow$ besoin de les synchroniser
  - outils de contrôle de version (CVS, Subversion)
  - outils pour PDA ou téléphones (HotSync)
  - logiciels de synchronisation de données (compte .Mac)
- Ces outils doivent être
  - simples à utiliser (éviter une fausse manœuvre de l'utilisateur)
  - corrects (pas de perte de donnée par un bug)

Cette présentation

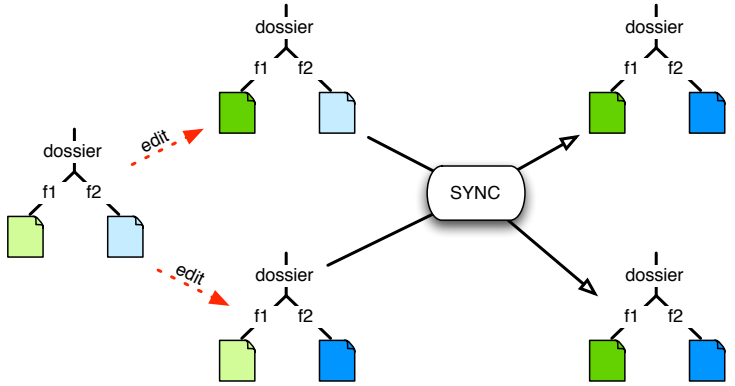
Deux outils de synchronisation, développés à l'Université de Pennsylvanie, par l'équipe de Benjamin Pierce

- Unison
  - Synchronisation de systèmes de fichiers
  - Multi-plateformes
  - Stable et utilisable
- HARMONY
  - Synchronisation de données structurées en arbres
  - En cours de développement

Démo

Un exemple simple

La synchronization devrait propager des modifications.



$\Longrightarrow$ Nécessité de détecter les modifications.

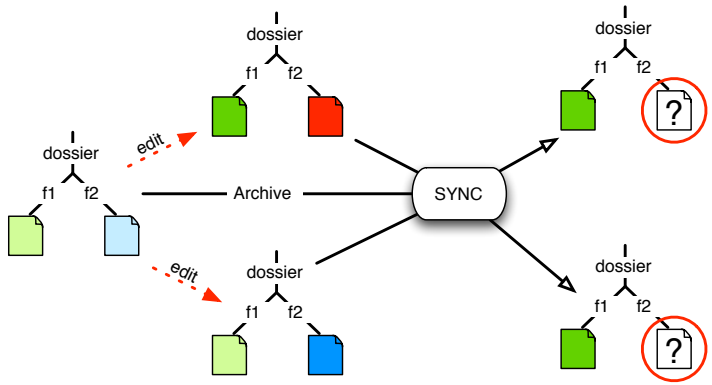
Détecter les modifications de fichiers

Une modification peut être vue comme

- Une conséquence d'une opération d'une application
  - Que faire si l'opération est annulée ensuite ?
  - Requiert la coopération de l'application
- Une conséquence d'une opération du système d'exploitation
  - Modification de méta données du fichier
  - Taille : non fiable
  - Heure de modification : non fiable (Excel peut changer le fichier sans changer l'heure)
- Une modification du contenu du fichier
  - Se rappeler de l'ancien contenu
  - Approche utilisée par Unison, qui stocke une signature de chaque fichier dans une archive

Un exemple de conflit

Modification concurrente du même fichier.



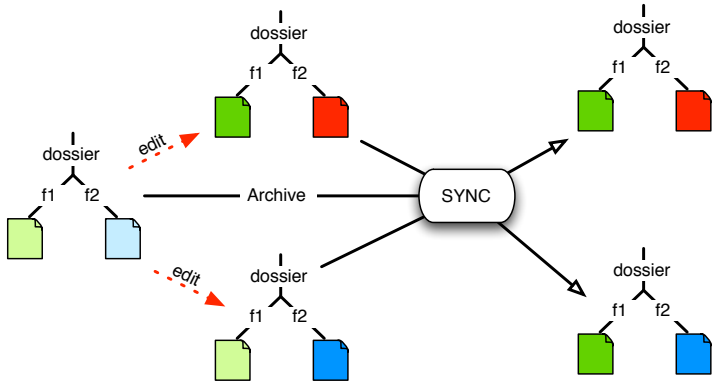
Conflit : modification d'un fichier dans les deux réplicas

Convergence ou Persistence ?

- Que faire lors d'un conflit ?
  - Fusionner les versions ou en choisir une arbitrairement
    - Garantit la convergence
    - Peut annuler une modification de l'utilisateur
  - Signaler le conflit sans changer les fichiers
    - Garantit la persistence des modifications
    - Les réplicas restent dans un état divergent
- Unison choisit la persistence
  - Utilisation automatique plus prévisible
  - Evite certaines surprises
    - Compiler un fichier en conflit CVS
    - Perte de données

La Persistence dans Unison

La synchronisation retourne deux fichiers différents



Formalisation de Systèmes de Fichiers

- Un chemin $p \in \mathcal{P}$ est une séquence de noms $p = x/y/\cdots/z$
- Système de fichier F : fonction totale de $\mathcal{P}$ vers $\mathcal{F} \cup \textsc{Dir} \cup \bot$
  - si l'élément au chemin p est un fichier, $F(p) \in \mathcal{F}$
  - si l'élément au chemin p est un dossier, $F(p) = \textsc{Dir}$
  - si le chemin p est absent de F, $F(p) = \bot$
- Contrainte : le parent d'un fichier ou dossier est un dossier

$$\forall p, x. \quad F(p/x) \neq \bot \implies F(p) = \textsc{Dir}$$

- Systèmes de fichiers différents si différents à un chemin

$$F \neq F' \iff \exists p.\, F(p) \neq F'(p)$$

  - Fichiers au contenu différents
  - Fichier ou dossier manquant
- Sous-arbre à p : A/p défini comme $(A/p)(p') = A(p/p')$

Fondations de Unison : Conflits

- Notion simple de conflit :
  - Identification des modifications grâce à une archive O
  - Modification du contenu d'un fichier des deux côté
  - Suppression d'un côté et modification de l'autre
- `conflict`(O, A, B, p) si et seulement si

	$A(p) \neq B(p)$	A et B sont différent à p
et	$A/p \neq O/p$	A a été modifié à ou sous p
et	$B/p \neq O/p$	B a été modifié à ou sous p

Fondations de Unison : Garanties

- On synchronise O, A, B en A', B', écrit $A', B' = \texttt{sync}(O, A, B)$
- Toute modification de l'utilisateur est préservée

$$A(p) \neq O(p) \implies A'(p) = A(p)$$
$$B(p) \neq O(p) \implies B'(p) = A(p)$$

- Aucune modification n'est "inventée"

$$A'(p) \neq A(p) \implies A'(p) = B(p)$$
$$B'(p) \neq B(p) \implies B'(p) = A(p)$$

- Aucune synchronisation sous un conflit

$$\texttt{conflict}(O, A, B, p) \implies A'/p = A/p \text{ et } B'/p = B/p$$

- Exemple de synchronisation correcte : $A, B = \texttt{sync}(O, A, B)$

L'algorithme

$A', B' = \texttt{sync}(O, A, B)$

- Si $A = B$: retourner A, B (réplicas égaux)
- Si $O = A$: retourner B, B (A inchangé)
- Si $O = B$: retourner A, A (B inchangé)
- Si A ou B ne sont pas des dossiers : conflit, retourner A, B
- Sinon (A et B dossiers) : récursion, pour tout nom n, $A'(n), B'(n) = \texttt{sync}(O(n), A(n), B(n))$, retourner A', B'

On montre que cet algorithme est correct et maximal (tout ce qui peut être synchronisé l'est)

Conclusions sur Unison

Cadre "simple" : synchronisation de systèmes de fichiers

- Fondation théorique précise
  - Spécification claire du comportement
  - Garanties de bon fonctionnement
  - Implémentation de référence prouvée correcte en Coq (Pierce et Vouillon, 2004)
- Implémentation robuste, portable, multi-plateformes
  - 23000 lignes de OCaml
  - Linux, Mac OS X, Windows, ...
  - Synchronisation hétérogène (Mac OS X $\leftrightarrow$ Windows, ...)
  - Nombreux cas particuliers à gérer
    - Noms de fichiers "bizarres" (avec accents, "\", "/", " :")
    - Métadonnées (date de modification, icône)
    - Liens symboliques

Harmony

Motivation : synchroniser plus que des systèmes de fichiers

- synchroniser le contenu
  - agenda, carnet d'adresses, signets, courriers électroniques ...
- synchroniser des formats *hétérogènes*
  - iCal avec Palm, Firefox et Safari ...

Ces informations peuvent être représentées par des arbres
Un système de fichier est un arbre $\implies$ généraliser l'algorithme

Des Fichiers aux Arbres

L'information dans un fichier n'est pas forcément présentée correctement

- fichier : suite d'octets
- interpréter cette suite d'octets (analyse lexicale, syntaxique)
- supprimer les informations inutiles
- réordonner la structure pour guider la synchronisation
  - synchronisation de la racine, puis récursivement de chaque branche indépendemment
  - problème de l'alignement : quel sous-arbre est synchronisé avec quel sous-arbre

On développe des lentilles pour ces transformations

Un carnet d'adresse simplissime

```
<vcard>
  <name>Lévy</name>
  <org>INRIA Rocquencourt</org>
  <email>Jean-Jacques.Levy@inria.fr</email>
</vcard>
<vcard>
  <name>Mauborgne</name>
  <org>ENS</org>
  <email>Laurent.Mauborgne@ens.fr</email>
</vcard>
<vcard>
  <name>Schmitt</name>
  <org>INRIA Rhône-Alpes</org>
  <email>alan.schmitt@m4x.org</email>
</vcard>
```

Abstraction en listes

$$\left[\begin{array}{lll}
\texttt{vcard} & \mapsto & \left[\begin{array}{lll}
\texttt{name} & \mapsto & \left[\texttt{Lévy} \mapsto [\right. \\
\texttt{org} & \mapsto & \left[\texttt{INRIA Rocquencourt} \mapsto [\right. \\
\texttt{email} & \mapsto & \left[\texttt{Jean-Jacques.Levy@inria.fr} \mapsto [\right.
\end{array}\right. \\
\texttt{vcard} & \mapsto & \left[\begin{array}{lll}
\texttt{name} & \mapsto & \left[\texttt{Mauborgne} \mapsto [\right. \\
\texttt{org} & \mapsto & \left[\texttt{ENS} \mapsto [\right. \\
\texttt{email} & \mapsto & \left[\texttt{Laurent.Mauborgne@ens.fr} \mapsto [\right.
\end{array}\right. \\
\texttt{vcard} & \mapsto & \left[\begin{array}{lll}
\texttt{name} & \mapsto & \left[\texttt{Schmitt} \mapsto [\right. \\
\texttt{org} & \mapsto & \left[\texttt{INRIA Rhône-Alpes} \mapsto [\right. \\
\texttt{email} & \mapsto & \left[\texttt{alan.schmitt@m4x.org} \mapsto [\right.
\end{array}\right.
\end{array}\right]$$

Abstraction en listes

$$\left[\begin{array}{lll}
\texttt{Lévy} & \mapsto & \left[\begin{array}{lll}
\texttt{org} & \mapsto & \left[\texttt{INRIA Rocquencourt} \mapsto [\right. \\
\texttt{email} & \mapsto & \left[\texttt{Jean-Jacques.Levy@inria.fr} \mapsto [\right.
\end{array}\right. \\
\texttt{Mauborgne} & \mapsto & \left[\begin{array}{lll}
\texttt{org} & \mapsto & \left[\texttt{ENS} \mapsto [\right. \\
\texttt{email} & \mapsto & \left[\texttt{Laurent.Mauborgne@ens.fr} \mapsto [\right.
\end{array}\right. \\
\texttt{Schmitt} & \mapsto & \left[\begin{array}{lll}
\texttt{org} & \mapsto & \left[\texttt{INRIA Rhône-Alpes} \mapsto [\right. \\
\texttt{email} & \mapsto & \left[\texttt{alan.schmitt@m4x.org} \mapsto [\right.
\end{array}\right.
\end{array}\right]$$

Clés de synchronisations

$$\left\{\begin{array}{l}\texttt{Lévy} \mapsto \left[\begin{array}{l}\texttt{org} \mapsto \left[\texttt{INRIA Rocquencourt} \mapsto [\right.\right.\\ \texttt{email} \mapsto \left[\texttt{Jean-Jacques.Levy@inria.fr} \mapsto [\right.\end{array}\right.\\ \texttt{Mauborgne} \mapsto \left[\begin{array}{l}\texttt{org} \mapsto \left[\texttt{ENS} \mapsto [\right.\right.\\ \texttt{email} \mapsto \left[\texttt{Laurent.Mauborgne@ens.fr} \mapsto [\right.\end{array}\right.\\ \texttt{Schmitt} \mapsto \left[\begin{array}{l}\texttt{org} \mapsto \left[\texttt{INRIA Rhône-Alpes} \mapsto [\right.\right.\\ \texttt{email} \mapsto \left[\texttt{alan.schmitt@m4x.org} \mapsto [\right.\end{array}\right.\end{array}\right.$$

Clés de synchronisations

$$\left\{\begin{array}{l}\texttt{Lévy} \mapsto \left[\begin{array}{l}\\ \texttt{email} \mapsto \left[\texttt{Jean-Jacques.Levy@inria.fr} \mapsto [\right.\right.\end{array}\right.\\ \texttt{Mauborgne} \mapsto \left[\begin{array}{l}\\ \texttt{email} \mapsto \left[\texttt{Laurent.Mauborgne@ens.fr} \mapsto [\right.\right.\end{array}\right.\\ \texttt{Schmitt} \mapsto \left[\begin{array}{l}\\ \texttt{email} \mapsto \left[\texttt{alan.schmitt@m4x.org} \mapsto [\right.\right.\end{array}\right.\end{array}\right.$$

La version à synchroniser

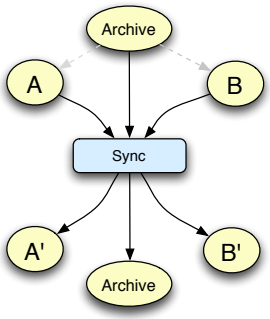
$$\left\{\begin{array}{l}\texttt{Lévy} \mapsto \left\{\texttt{Jean-Jacques.Levy@inria.fr} \mapsto \{\right.\\ \texttt{Mauborgne} \mapsto \left\{\texttt{Laurent.Mauborgne@ens.fr} \mapsto \{\right.\\ \texttt{Schmitt} \mapsto \left\{\texttt{alan.schmitt@m4x.org} \mapsto \{\right.\end{array}\right.$$

Des Lentilles Bidirectionnelles

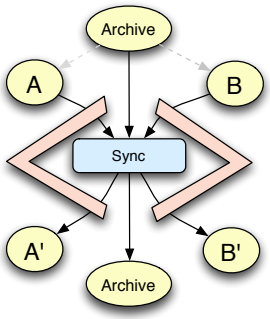
- Lentille : passe de la version concrète à la version abstraite
- Synchronisation des versions abstraites
- Nécessité créer de nouvelles versions concrètes

Lentilles : transformateurs d'arbres bidirectionnels

Unison



Harmony



Harmony

- Langage de transformateurs d'arbres bidirectionel : Focal
  - Primitives prouvées bidirectionnelles
  - Constructeurs pour les assembler produisant des lentilles bidirectionnelles par construction
- Moteur de synchronisation générique
  - Peut prendre en compte le schéma (forme finale attendue)
- Implémentation en cours
  - 7600 lignes de OCaml, 700 lignes de Focal
  - Benjamin Pierce, Nate Foster, Stéphane Lescuyer (X02), Kim Thang Nguyen (X02), Alan Schmitt
- Nombreux exemples
  - Signets, agendas, carnet d'adresses (en cours), fichiers BibTeX
- Deux utilisateurs

Conclusion

- Synchronisation : problème simple, riche, courant
- Spécification du comportement attendu crucial
  - Outil prévisible
  - Vérification de la correction de l'outil
- Utilisation de techniques des langages de programmation
  - Harmony : synchronisation générique, spécialisation par développement de lentilles
  - Approche modulaire

Le Mot de la Fin

- Perdre une Ariane V : 150 millions d'euros

Le Mot de la Fin

- Perdre une Ariane V : 150 millions d'euros
- Perdre un Airbus A380 : 200 millions d'euros

Le Mot de la Fin

- Perdre une Ariane V : 150 millions d'euros
- Perdre un Airbus A380 : 200 millions d'euros
- Perdre ses photos de vacances : ça n'a pas de prix

`http://www.cis.upenn.edu/~bcpierce/unison/`

`http://www.cis.upenn.edu/~bcpierce/harmony/index.html`