Architecture de sauvegarde, le retour

Ludovic Courtès

LAAS-CNRS

- **Découpage et sauvegarde d'un fichier**
  - Introduction
  - Découper et indexer une séquence d'octets
  - Structure de méta-données : *Merkle trees* (rappel)
  - Découpage : choix d'algorithmes
  - Indexation/identification des blocs
  - Une solution : indexation fonction du contenu
  - Chiffrement au niveau bloc
  - Découpage et indexation d'une séquence d'octets : implémentation

- Gestion des méta-données de haut-niveau
- Déroulement de la sauvegarde

Introduction

- **Une approche par étapes**

  1. découpage et indexation d'une **séquence d'octets**
  2. … de l'association entre un **nom de fichier** (+ autres méta-données) et un **index**
  3. … d'une **liste des fichiers** (leurs indexes)
  4. … d'un **point d'entrée pour cette sauvegarde** (+ date, etc.)

- **Principes**

  - sauvegarder **tout l'état nécessaire à la restauration** (données + méta-données) à distance
  - modularité, flexibilité, simplicité
  - séparation des préoccupations

Découper et indexer une séquence d'octets

- **Étapes**

  1. **découper** en blocs
  2. **assigner un identifiant** à chaque bloc
  3. créer des **méta-données séparées** permettant de ré-assembler les blocs

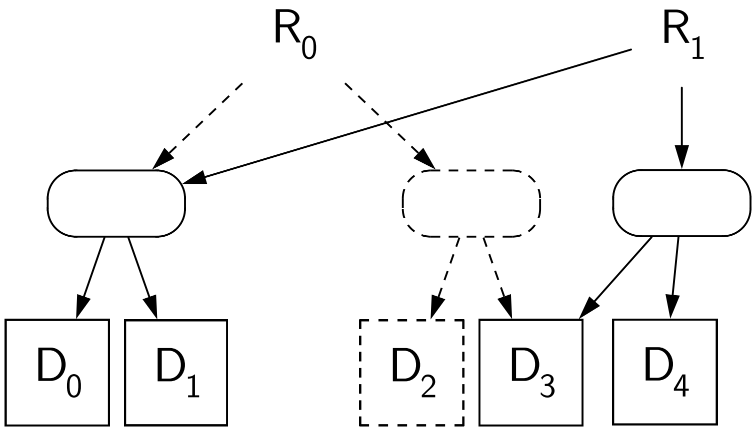
- **Choix du format des méta-données**

  - séparation des préoccupations
  - séparation $\Longrightarrow$ possibilité de **partage des blocs de données**
  - traitement des méta-données **comme des données** (e.g. réplication)

Structure de méta-données : *Merkle trees* (rappel)

- Fichier découpé en blocs : D_0 à D_n
- Création de *blocs d'indexes* intermédiaires
- L'index de la **racine** (R_0) suffit pour récupérer tous les blocs

⇒ Structure **récursive**
⇒ Permet le **partage de parties de méta-données**



Découpage : choix d'algorithmes

- **Découpage en blocs de taille fixe**
  - simple, peu coûteux en CPU
- **Découpage fonction du contenu (cf. Udi Manber)**
  - donne des **blocs de taille variable**
  - permet de **repérer des parties identiques** entre deux séquences d'octets
  - ... et donc d'**économiser l'espace**

$\Rightarrow$ Évaluation en cours.

Indexation/identification des blocs

⇒ **Comment référencer un bloc ?**

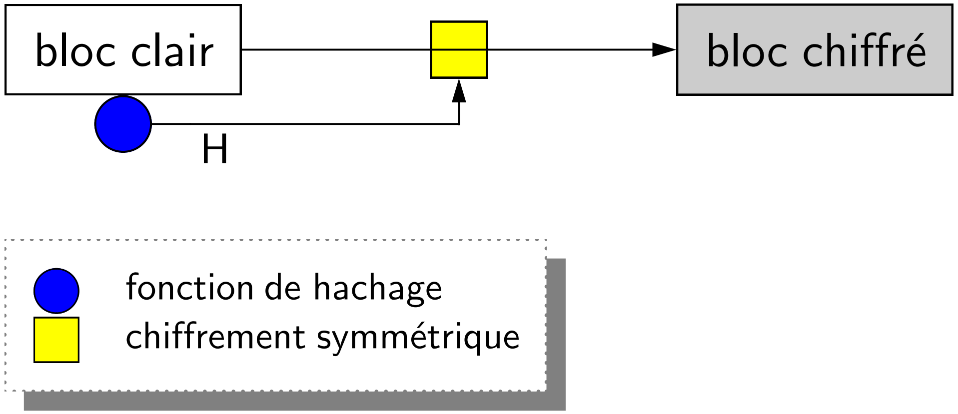
- **Problèmes**
  - quantité d'information à conserver localement ? (idéalement : rien)
  - travail demandé aux contributeurs ?
  - validité dans l'espace ? dans le temps ?
  - risques de collisions ?

Une solution : indexation fonction du contenu

- **espace de nom universel** : chaque identifiant de bloc est considéré comme **globalement unique**
- les méta-données peuvent être **calculées une fois pour toutes**
- identifiants **uniques dans le temps** : après un crash, possibilité de continuer à utiliser ces identifiants
- facilite la **vérification d'intégrité** d'un bloc
- sémantique « **ajout seulement** » : une nouvelle version d'un bloc ne vient jamais remplacer une précédente $\Rightarrow$ **pas de gestion de cohérence requise**
- identification de blocs identiques $\Rightarrow$ **économie d'espace disque**

Chiffrement au niveau bloc

- chiffrer les blocs individuellement $\Longrightarrow$ permet de **conserver la relation d'égalité** entre blocs
- une possibilité : **chiffrement convergent**



Découpage et indexation d'une séquence d'octets : implémentation

- Une bibliothèque (`libchop`)
- Et un outil compagnon (`chop-archiver`)

```
$ chop-archiver -R host.example.com --archive britney.mp3
SHA1:34eec486859b91f992928e2d64a6441e3773ea50/82
```

```
$ chop-archiver --cipher blowfish -R host.example.com \
                --archive britney.mp3
SHA1:51c1ff578a851e9c48ef1e90dbbeb064d64ff13a,SHA1:18837c0c9bd7d9f81975ea3aa0d1b734/d2
```

condensé du bloc chiffré

clef (condensé du clair)

taille

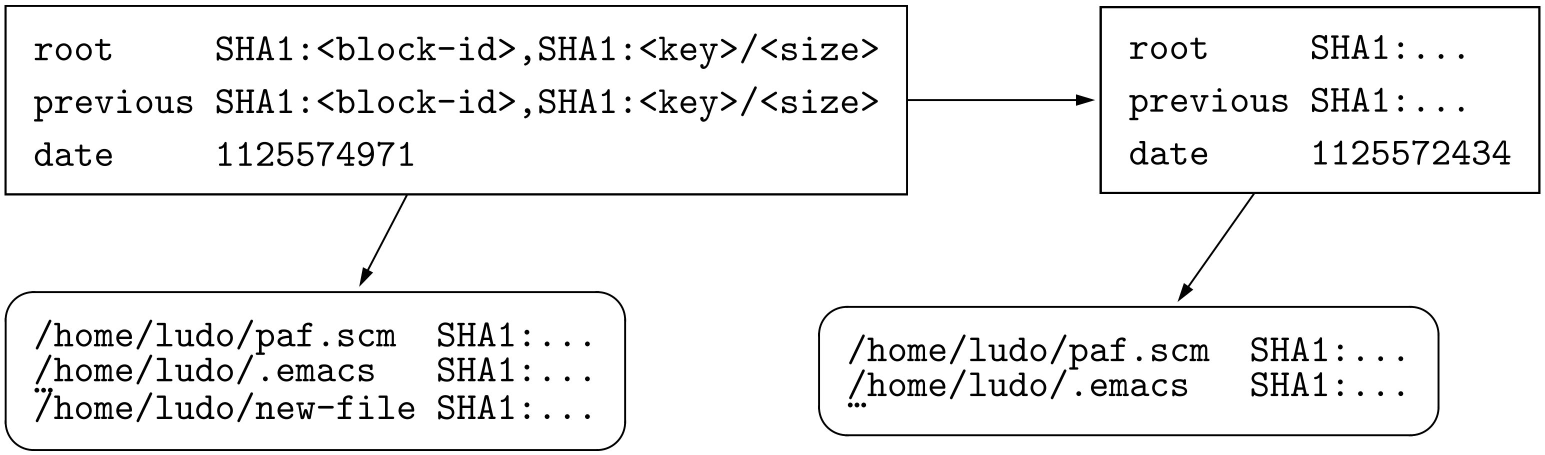
- Découpage et sauvegarde d'un fichier
- **Gestion des méta-données de haut-niveau**
  - Les méta-données d'un fichier
  - Proposition : répertoire et chaîne d'indexes
- Déroulement de la sauvegarde

Les méta-données d'un fichier

- **Les méta-données intéressantes**
  - le nom du fichier
  - date ou version
  - propriétaire, attributs étendus, etc.
- **En tant qu'« objets de première classe »**
  - ces méta-données sont une **séquence d'octets**
  - utilisation des techniques **vues précédemment**

Proposition : répertoire et chaîne d'indexes

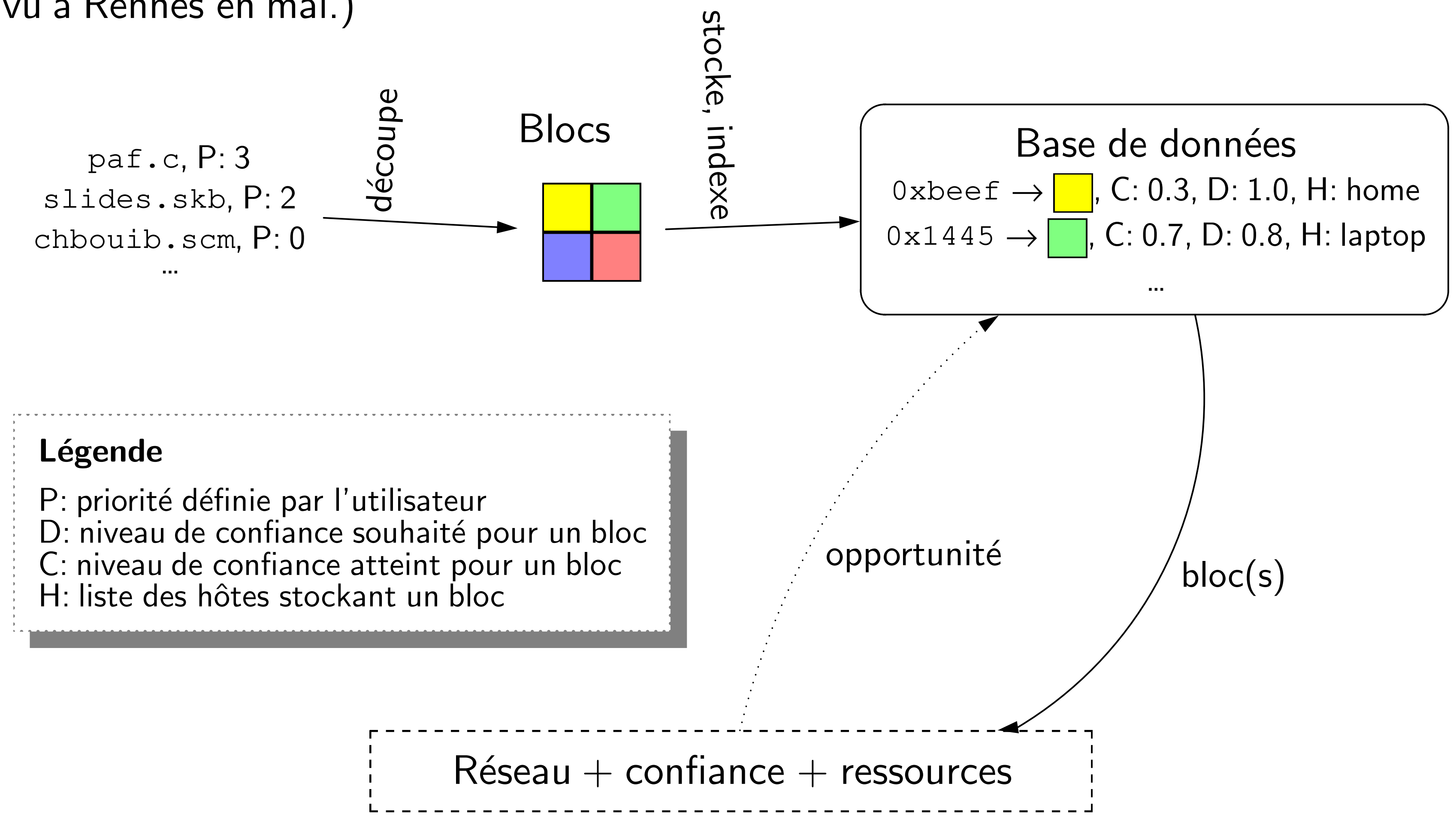
- **Répertoire d'indexes** : contient une liste d'associations nom/clef
- **Chaîne d'index** : contient l'index de l'élément précédent dans la chaîne et une date



- Découpage et sauvegarde d'un fichier
- Gestion des méta-données de haut-niveau
- **Déroulement de la sauvegarde**
  - Sauvegarde : vue détaillée
  - Sauvegarde : début d'implémentation

Sauvegarde : vue détaillée

(Déjà vu à Rennes en mai.)



Sauvegarde : début d'implémentation

Découpage et indexation

1. liste de fichiers définie par l'utilisateur
2. attente de nouveaux fichiers/fichiers modifiés
3. **découpage/indexation** des dits fichiers
4. découpage/indexation des méta-données
5. insertion des blocs **dans une file d'attente locale**
6. pour chaque bloc : maintien d'un « **compteur de références** »

Émission des blocs

- pour l'instant : régulièrement, sur des machines données
- futur : réagir aux **évènements réseaux**, **découvrir** des contributeurs, etc.
- pour chaque bloc : maintien d'une **liste des répliques**

Fin

Questions ?