# Éléments de comparaison de systèmes de gestion de code source

Matthieu Herrb

CNRS LAAS

COMPIL, 10 mars 2008

# Agenda

# Agenda

# Trois modèles de fonctionnement

## Local

Fonctionne dans un système de fichiers local. Pas de réseau.

- SCCS, RCS,...

## Client/Serveur (ou centralisé)

Un serveur centralise le dépot, accessible à distance.

- CVS
- Subversion

## Distribué

Multiples copies du dépots, branches locales.

- bitkeeper, monotone, arch, darcs
- mercurial, git, bazaar

# Critères de choix

- Modèle de développement (Centralisé ou non)
- Souplesse d'utilisation :
  - gestion des branches (fusion)
  - déplacement/renommage des fichiers
- Sécurité
  - intégrité : signature des fichiers, des commits
  - gestion explicite des droits d'accès par utilisateur
  - disponibilité : possibilité de récupération des données en cas de corruption du dépot
- Efficacité, Vitesse - important pour des gros projets
- Diffusion, développement actif
- Portabilité (Mac OS X/Unix/Windows).

# Agenda

# SCCS, RCS

## SCCS

SCM historique d'Unix.
défini les concepts de base de beaucoup de SCMs.
Pas Open Source
Verrouillage très strict $\longrightarrow$ pas de fusion.

## GNU RCS

Extension de SCCS. Introduit la notion de fusion.
Gestion des branches très lourde.

Principales limites : un seul utilisateur à la fois, une seule copie de travail.

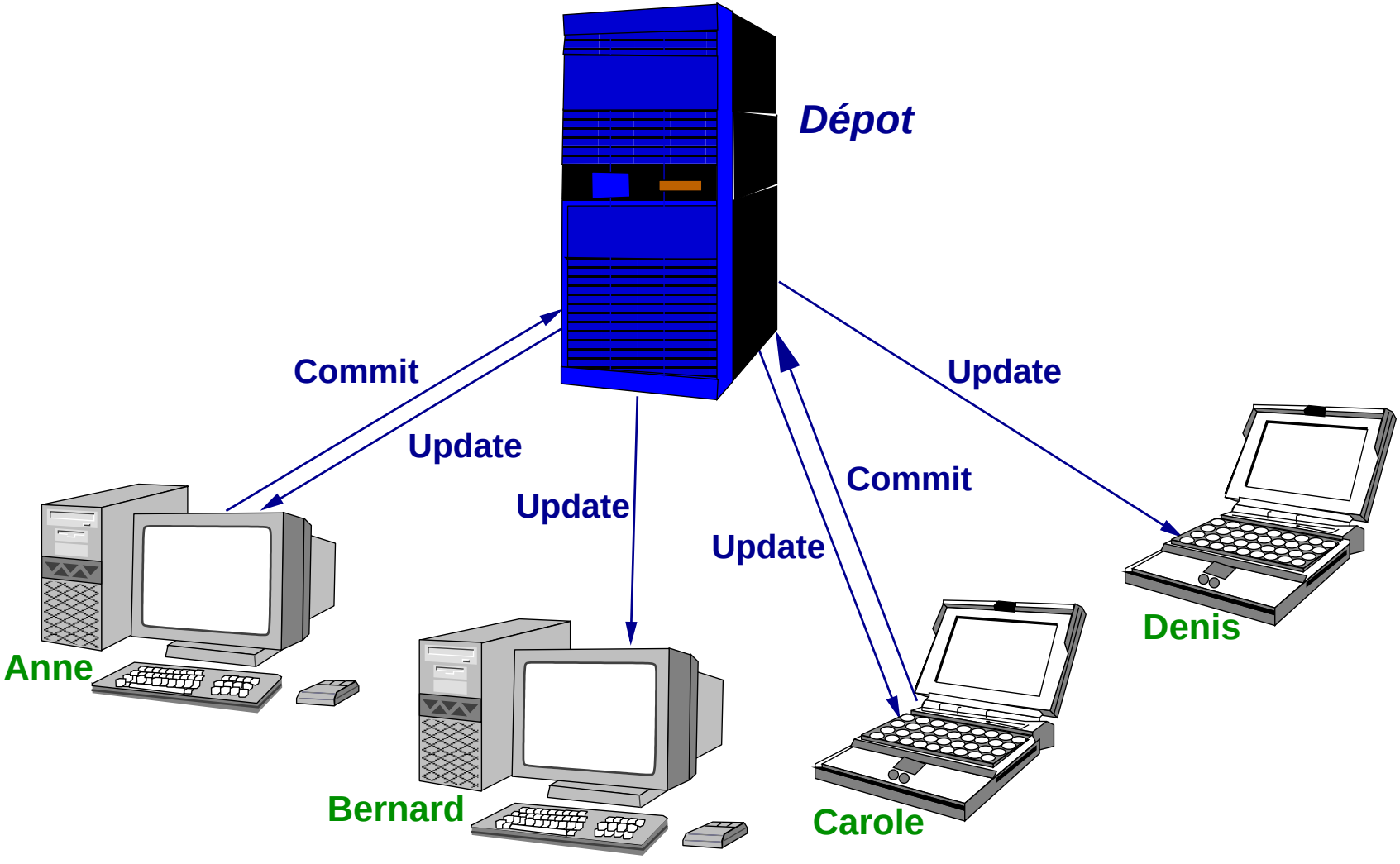
Reste intéressant pour certains cas simples.

# Agenda

# Client-serveur - principe

- Dépot stocké dans un endroit partagé
  - par le système de fichiers
  - par un mécanisme réseau (rsh/ssh ou protocole dédié)
- Plusieurs copies de travail en parallèle : opérations de fusion.
- Nécessaire d'avoir la connexion au dépot pour committer.
- Le «tronc» a une importance particulière : modèle très centralisé.

# Client-serveur - principe



# CVS

**C**oncurrent **V**ersion **S**ystem
`http://www.nongnu.org/cvs/`

- Basé sur RCS. Centralise le dépot et autorise plusieurs copies de travail concurrentes.
- Initialement uniquement local dans un système de fichiers.
- Mode client/serveur simple
- Ne gère pas l'authentification - nécessite un compte Unix par utilisateur sur le serveur
- Travaille fichier par fichier. Pas de commits atomiques, ne gère pas les rennomages ou les déplacements de fichiers
- Notion de "vendor branch"
- Gestion des branches très lourde. Pas adaptée pour des développements parallèles.

# OpenCVS


Réimplémentation de GNU CVS sous licence BSD.

- Compatible au niveau dépot, commandes, protocole.
- Meilleur contrôle de la sécurité.
- Extensions prévues : commits atomiques, support du renommage.

Code pas encore utilisable en production.

# Subversion

`svn`
`http://subversion.tigris.org/`


- «Successeur» de CVS
- Interface utilisateur similaire à CVS
- Commits atomiques
- Gère le renommage de fichiers
- Gère les meta-données (droits d'accès, propriétaire)
- Meilleure gestion des branches, mais pas de mémoire des fusions
- Stockage sophistiqué (base de données,...)
- Accès distants via HTTP/DAV

# Perforce

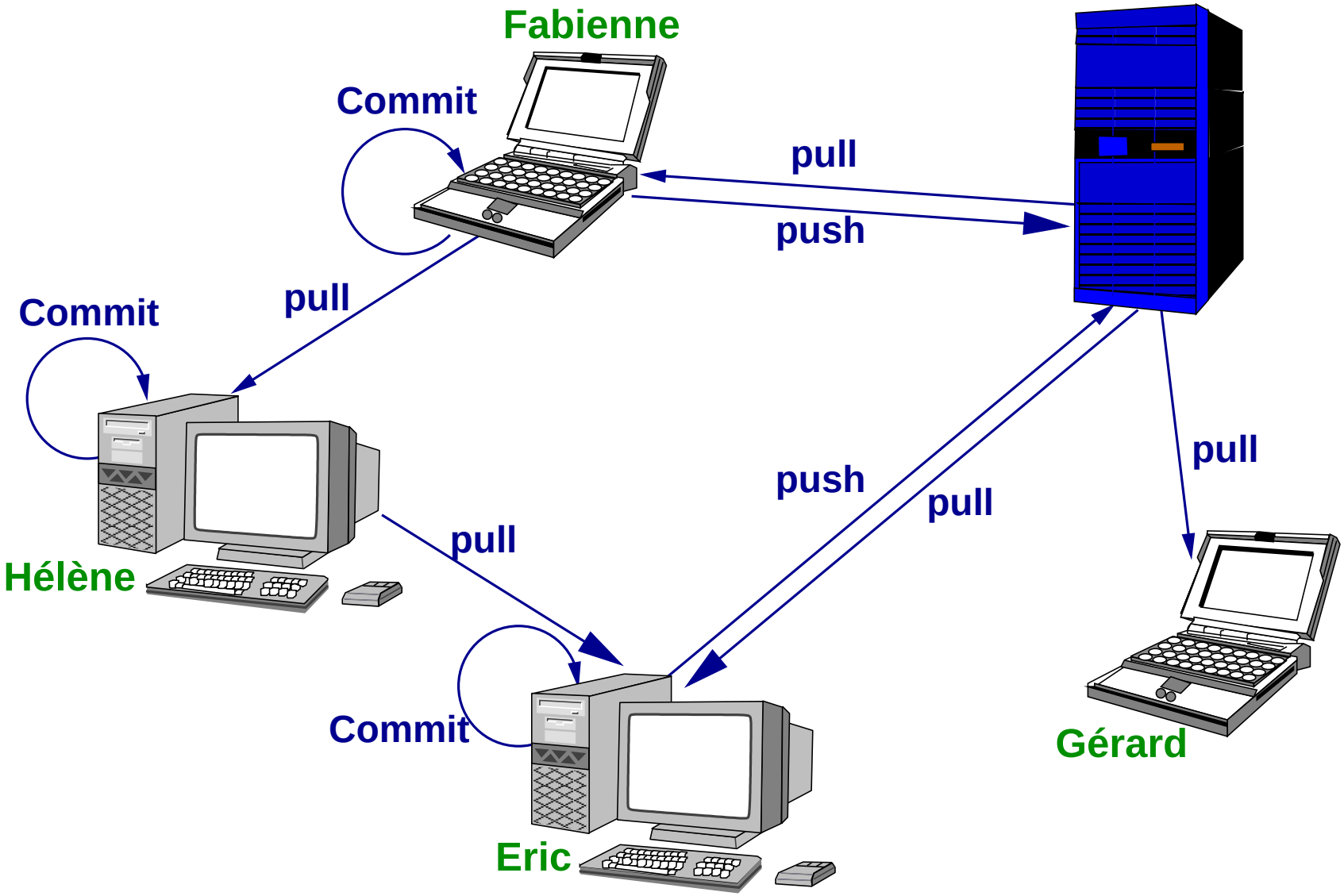

- Propriétaire
- Licences gratuites pour projets Open Source (FreeBSD, Perl, par exemple).
- Rapide
- Branches faciles et peu coûteuses
- Bon algorithme de fusion

# Agenda

# Systèmes distribués - principe

- Plus de dépot centralisé
- Chaque développeur a sa copie avec ses branches privées
- Opérations push/pull : synchronisation avec les autres dépots.
- Simplification de la fusion de branches en gardant l'historique des fusions.
- Influence sur la philosophie de développement : plus de liberté,
  mais risque de dispersion...

# Systèmes distribués - principe



# Le premier SCM distribué : BitKeeper

- Développé par Larry McVoy (Sun, Sgi,...) pour le noyau Linux.
- Basé sur SCCS en ajoutant la notion de réplication du dépot et le suivi des méta-données.
- Le premier SCM à utiliser une copie du dépot par branche.
- Produit non libre avec une licence très restrictive.
- Abandonné par Linux après le fiasco.

# git

`http://git.or.cz/`


Développé par Linus Torvalds pour remplacer BitKeeper.

- Architecture à deux couches :
  - la plomberie : un système de fichiers adressable par le contenu (via hashs SHA-1) avec gestion de l'historique.
  - la porcelaine : les commandes de haut-niveau destinées à l'utilisateur normal de git. très proche de Mercurial.
- Utilisation systématique de crypto. Un objet qui rentre dans git est immuable.
- Très rapide.
- Utilisé par de nombreux projets : Linux, X.Org, Mesa3d, ...
- Développement très actif.
- N'aime pas Windows.

# Mercurial

`hg`
`http://www.selenic.com/mercurial/`


- Écrit en python
- Utilise des hash SHA-1
- Relativement compact
- Extensible (framework pour des extensions)
- Développement actif
- Utilisateurs : OpenSolaris, Xen
- Projet encore jeune.
- Gestion des renommages par copie.

# Baazar

`bzr`
`http://www.bazaar-vcs.org/`

- Réécrit en python
- Sponsorisé par Cannonical (Ubuntu)
- Branches externes (utilisant d'autres SCM) (bientôt...)
- Utilisateurs : Ubuntu launchpad, Drupal, Samba

# Agenda

# Conclusion

- Technologie en pleine (r)évolution.
- L'apparition depuis 2002 des SCM distribués change la façon de travailler.
- Attention à la pérénité.

# Questions ?

# Le fiasco de BitKeeper

- Linux Torvalds a commencé à utiliser BK vers 2002 (auparavant il n'utilisait pas de SCM pour Linux!)
- De nombreux mécontents (à cause de la licence), encourageant le développement d'autres SCM distribués (Arch, Darcs, Mercurial)
- En 2005, Andrew Tridgell (Samba) commence le développement d'un client libre compatible BK. Larry McMoy révoque toutes les licences gratuites de BK. Linux n'a à nouveau plus de SCM.
- Linus teste les SCM distribués existants. Aucun ne convient à ses besoins (soit trop limités, soit trop lents, soit trop complexes)

$\longrightarrow$ Linus commence alors le développement de son propre SCM distribué : **git**.