Servicii de fisiere in retea si internet

Master IISC
Ciurea Catalin

Cuprins

- Sisteme de control al versiunilor fisierelor
  - Concepte. Descriere.
  - Descriere sistemul Subversion
  - Sisteme distribuite de versionare.
- Protocolul *Rsync* pentru sincronizarea fisierelor
  - Descriere algoritm rsync
  - *Checksum. Rolling checksum.*

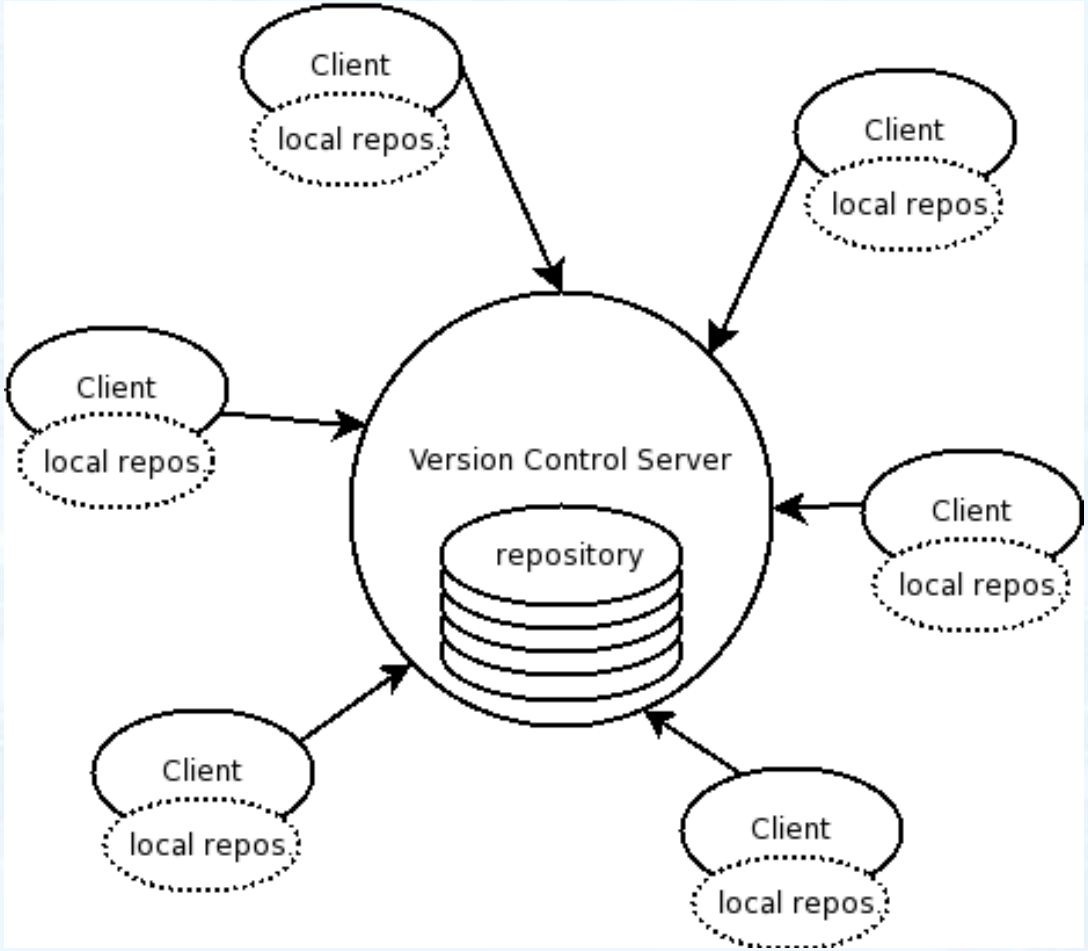
Versionarea fisierelor

- Managementul schimbarilor oricarei informatii stocate sub forma fisierelor
- Sistemele de versionare mentin toate versiunile pe care le are un set de fisiere pe tot parcursul existentei pe HDD
- Se poate reveni oricand la orice versiune
- Sunt folosite intensiv in dezvoltarea de proiecte(mai ales software) de catre echipe dispersate.

Versionarea fisierelor

- In orice domeniu in care informatiile se schimba des este loc pentru versionare.
- Schimbarile intr-un set de fisiere sunt identificate printr-un nr. sau un cod.
  - Exemplu: *revision 1, revision 2*
- Versiunile sunt marcate cu etichete de timp(timestamp) si pot fi comparate, restaurate si pentru unele tipuri de fisiere combinate intr-un singur fisier cu ambele modificari.

Structura generala a unui sistem de versionare



Exemplu: Subversion

- Aparut in 2000, iar in prezent este integrat in comunitatea open-source.(proiect de top al organizatiei non-profit Apache)
- Sistem de versionare centralizat (paradigma client-server)
- Mentine setul de fisiere intr-o locatie denumita repository accesibila prin internet
- Repository – mentine orice schimbare efectuata asupra fisierelor

Arhitectura.Modele de implementare

- *Lock-modify-unlock:*un singur utilizator poate modifica un fisier la un moment dat.
- Utilizatorul pune un lacat pe fisier.
- Ceilalti nu pot aduce modificari pana cand lacatul nu este eliberat.
- *Probleme*:
  - este prea restrictiv: lacatul poate fi uitat
  - poate crea serializari fara necesitate

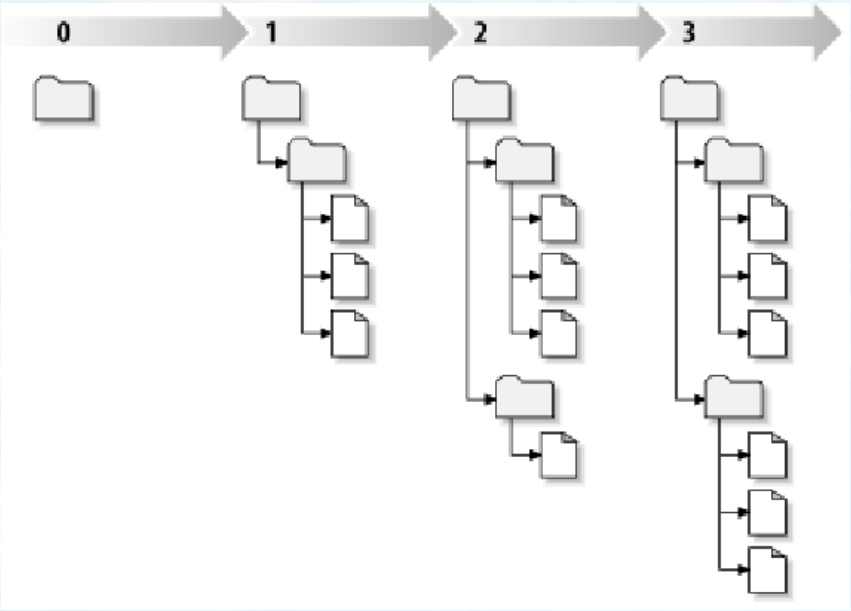
Modelul de implementare Subversion

- *Copy-modify-merge:* fiecare user isi creaza o copie locala
- Userii lucreaza in paralel pe copiile lor.
- In final copiile sunt combinate (*merge*).
- Cineva trebuie sa se asigure ca migrarea se face corect.
- Problema frecventa: 2 useri modifica aceeasi zona de cod → conflict.

Modelul de implementare Subversion

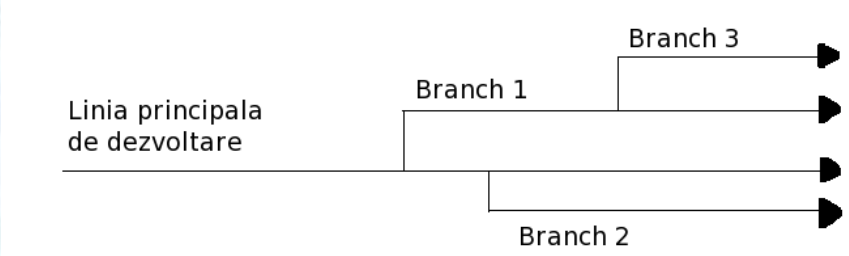
- Solutia *copy-modify-merge* desi pare haotica, in practica ruleaza foarte eficient.
- Timpul necesar rezolvarii conflictelor este mai mic comparativ cu timpul necesar rezolvarii blocajelor.
- Factor important: comunicarea intre useri
- Desi sistemul cu blocaje pare sa evite conflictele, de fapt reduce productivitatea.

Arhitectura repository-ului



- Fiecare modificare a unui fisier creeaza o noua versiune a intregului arbore.
- Fiecare versiune va avea propria sa radacina.
- Ptr. Eficienta sistemul mentine doar ***diferentele*** intre fisiere.
- Deci sistemul consuma spatiu prop. cu nr. de modificari si nu cu nr. total de versiuni.

Branches



- Branch: linie de dezvoltare independenta dar care a plecat in trecut de la o versiune comuna.
- Nu exista un concept special de branch: acesta exista cu scop organizatoric.
- Cand un branch se creeaza se copiaza structura repository-ului intr-un director special *Branches*.

Tags.Trunks.

- *Tag*:stare fixa(*snapshot*) a structurii repository-ului la un moment dat.
- Sunt versiuni care nu se doresc a fi modificate pe viitor.
- *Trunk*: notiunea de trunk refera mereu ultima versiune a fisierelor din repository.
- Orice noua versiune a ierarhiei va fi localizata in directorul *Trunk*.

Probleme si limitari ale sistemului Subversion.

- Lipsa unor facilitati avansate pentru managementul repository-ului.
- Sistemul mentine fisiere ascunse si local pentru sincronizare.
  - Poate fi o problema in proiecte mari sau in cazul in care se lucreaza pe mai multe *branch*-uri simultan.
- Nu pastreaza datele modificarii fisierelor in sistem ci datele actualizarii in repository.
  - De multe ori se doreste data modificarii locale in sistem

Sisteme distribuite de versionare

- Inovatie recenta in acest domeniu(*software revision control*).
- Abordare *peer-to-peer:* toti participantii sunt atat consumatori cat si furnizori de resurse.
- Vine in contrast cu paradigma client-server: serverele sunt furnizorii, iar clientii sunt consumatorii.

Sisteme distribuite de versionare

- Nu se utilizeaza un *repository* central.
- Copiile userilor sunt si *repository*-uri.
- Sincronizarea se face facand schimburi de *patch*-uri.(*patch* -> set de modificari).
- Patch-ul poate fi aplicat unui fisier pentru a aplica modificarile incluse.

Caracteristici sisteme de versionare distribuite.

- Nu exista versiune de referinta a fisierelor in cauza. Versiunile dispersate la useri.
- Operatii uzuale: actualizari, reverse, vizualizarea istoriei sunt rapide.
- Comunicatia necesara doar cand se fac actualizari in celelalte *peer*-uri.
- Fiecare copie locala este un fel de *back-up* remote asigurand o securitate nativa.

Avantaje oferite de sistemele distribuite de versionare

- Permite userilor sa lucreze productiv chiar daca acestia nu sunt conectati la internet.
- Permite implicarea in proiecte fara permisiuni de la administratorul repository-ului.
- Permite lucrul privat cu versiunile care nu se doresc a fi impartite.
- Evita nevoia utilizarii unui server central.

Protocolul rsync ptr. sincronizarea fisierelor

- Oglindeste ierarhii de fisiere intre doua locatii.(*mirroring*)
- Sincronizeaza fisiere sau directoare minimizand transferul de date prin aplicarea compresiei Delta.
- Compresie Delta – trimiterea datelor sub forma diferentelor(*data differencing*).
- Protocolul este implementat intr-un software ptr. sistemele UNIX(*rsync*).

Protocolul rsync

- Poate sincroniza fisierele in mod nativ rsync sau prin intermediul protocoalelor RSH(*remote shell*) sau SSH(*secure shell*).
- In mod *daemon* asculta pe portul 873.
- Este utilizat intensiv in prezent pentru sincronizarea unor seturi de fisiere.
- Este distribuit sub licenta GNU, deci este un software *open source*.

Algoritmul rsync (1)

- Problema care se pune este actualizarea unui fisier remote B la structura unui fisier A local.
- Operatia de compresie si apoi transmisie ineficienta in cazul fisierelor mari.
- O solutie este trimiterea diferentelor, dar pentru calcularea setului de diferente(*patch*) este nevoie de prezenta ambelor fisiere local.

Algoritmul rsync (2)

- Algoritmul rsync calculeaza eficient care din partile unui fisier local sunt echivalente cu partile unui alt fisier destinatie.
- Aceste parti nu trebuiesc trimise prin retea la destinatie.
- Pentru a minimiza si mai mult cantitatea de informatie transmisa se poate folosi si un algoritm uzual de compresie.

Algoritmul rsync (3)

- Sistemul destinatie isi imparte fisierul in blocuri fixe de dim. S si calc. ptr. fiecare bloc 2 *checksum*-uri:unul mai “slab” pe 32 biti si altul pe 128 biti de tip MD4.
- *Checksum:* functie care calculeaza o secventa de octeti **unica** in functie de secventa de la intrare.
- Sistemul destinatia trimite cele 2 *checksum*-uri la sistemul sursa.

Algoritmul rsync (4)

- Sursa va cauta in fisierul propriu toate blocurile de lungime S(**cu orice offset**) care au aceleasi checksum-uri precum cele primite.
- Se face foarte eficient intr-o singura trecere folosind o proprietate a *checksum*-urilor.
- Rolling checksum: se calculeaza rapid *checksum*-urile indiferent de offset folosind formule recurente de calcul.

Algoritmul rsync (5)

- Sursa trimite la destinatie un set de instructiuni pentru construirea la destinatie a fisierului aflat la sursa.
- Fiecare instructiune este o referinta la un bloc de date sau o secventa de date in cazul blocurilor fara corespondent la destinatie.

Soft-ul rsync

- Aplicatia in sine rsync are incorporata pe langa protocolul rsync si cateva facilitati importante:
  - compresia si decompresia blocurilor de date
  - suport ptr. protocolul ssh ptr. criptarea informatiilor
  - poate limita banda consumata (facilitate utila suportata de putine alte protocoale.)

Va Multumesc