Hierarchical Distributed Repositories in CVS

Miroslav Jurišić

January 31, 2001

Brief history of CVS

- Revision Control System (RCS, 1982)
- CVS shell scripts (1990)
- CVS in C
- Client-server CVS

Motivation

- Repository is the unit of management
- Large projects need distributed management
- Scenario 1: private modifications to a public project
- Scenario 2: public modifications to a public project
- Scenario 3: public and private repositories for a project
- Scenario 4: pristine repositories for a project
- Scenario 5: off-line operation

Existing Repository Model

- Client: working files
  - Per-directory administrative information: repository location
  - Per-file administrative information: checkout revision, date
- Server: repository
  - Revision history of every file
  - Revisions organized in a tree
  - Symbolic tags

Existing Repository Model

- Client-server protocol
  - Authentication: connection port depends on authentication method
  - Client requests regenerate working files on the server
  - Server executes commands locally and relays output back to the client
- User interface
  - Translates user requests to client requests
  - Minimal translation for command line clients

Desired Repository Model

- Full generality is too complicated
- Typical way people use CVS
  - Most operations are performed on a single branch
  - Cross-branch operations are rare
- Simplify the problem by requiring every branch to be stored on one server
- Revision hierarchy parallels server hierarchy – hierarchically distributed repository

Modifications To the Repository Model

- Client: working files
  - Per-file repository location
- Server: repository
  - Parent branch location
  - Child branch locations at branchpoint
- User interface
  - New syntax for `tag` for remote branch creation

Modifications To the Repository Model

- Client-server protocol
  - Most requests and responses unchanged, because they never operate on more than one version of a single file
  - Authentication: consolidated authentication negotiation on one port
  - `Not-Carried` returned when a needed revision is not available to the server
  - `Remote-Revision` used to provide a needed revision to the server
  - `tag` used to create new remote branch points: new syntax, new response: `Create-remote-branch`
  - `add` used to create new remote branches: new syntax

Further Work

- Make `update` work across servers
- Coalesce multiple requests to a single server into one session
- Better error handling: some errors are fatal when they don't need to be
- Client-side caching of repository hierarchy
- Submit the changes to CVS maintainers

Conclusion

- Gnu/Cyclic CVS implementation is painful
- Ideas and intents of distributed repositories are feasible and useful
- Open source development decentralized, needs decentralized project management
- Consider distributed repositories for inclusion in the emerging new version control systems