DVCS or a new way to use Version Control Systems for FreeBSD

Ollivier ROBERT
<roberto@FreeBSD.org>

BSDCan 2006
Ottawa, Canada
May, 12-13th, 2006

Agenda

The ancestors: SCCS, RCS

- File-oriented
- Use a subdirectory to store deltas and metadata
- Use lock-based architecture
- Support shared developments through NFS (fragile)
- SCCS is proprietary (System V), RCS is Open Source
- a SCCS clone exists: CSSC
- You can have a central repository with symlinks (RCS)

CVS, the *de facto* VCS for the free world

- Initially written as shell wrappers over RCS then rewritten in C
- Centralised server
- Easy UI
- Use *sandboxes* to avoid locking
- Simple 3-way merges
- Can be replicated through *CVSup* or even `rsync`
- Extensive documentation (papers, websites, books)
- Free software and used everywhere (SourceForge for example)

CVS annoyances and flaws

BUT...

- No atomic commits so no changesets
- No support for renames
- No real handling of permissions/ownership and directories
- Access control is primitive, relying on the underlying filesystem and UNIX user model
- Branching is weak (no memory of branching/merging)
- Support for third party code is weak (vendor branch)
- Backend file format (inherited from RCS) is sub-optimal

CVS Next generation: Subversion

Written by former CVS authors to be *CVS done right*:

- Centralised architecture
- UI is very close to CVS
- Atomic commits
- Renaming of directories/files
- Can control metadata (ownership, permissions)
- Easy and cheap branches & tags
- Several storage backends (BDB, FSFS)
- Remote access through Apache/WebDAV/SSH
- Documentation available (websites, books)

SVN flaws

BUT...

- No memory of merging either
- Centralised design
- File rename is copy+delete: seen as worse than CVS by some
- Big piece of software (apr, apr-util, . . .)
- Metadata overhead on the client side is rather heavy (like 5 files per file)
- Not very network-efficient in some cases

CVS Next generation: Perforce

- Centralised design
- Proprietary and closed source software
- Open Source friendly with free licenses for FOSS projects – FreeBSD, Perl for example
- Fast and easy to install
- Easy and cheap branching
- Good merging algorithm

First Distributed VCS: BitKeeper

- Created by Larry McVoy (BitMover, Inc.) with the Linux kernel as proving ground
- Based on SCCS architecture extended to allow replication of repositories and metadata tracking
- First to have a *repository per branch* design
- Closed source product with a very restrictive license
- ~~Free license to develop free software (with constraints) and a "Don't compete" clause~~.

The competitors: Arch, Darcs, Monotone

Arch	One of the first DVCS. First in Shell scripts/C (`larch`) and later rewritten in C (`tla`). Forked as `bazaar` by Canonical, Ltd. and former contributors to `tla`. Complicated UI, very slow on big trees.
Darcs	Simple UI. Implemented in Haskell. Small, rather slow and likes to have everything in memory. Can manage multiple hunks within a diff.
Monotone	Uses crypto hashes to represent revisions. Uses certificates and crypto everywhere. Still slow and can't handle repositories such as the Linux kernel.

BK & the Linux debacle

- L. Torvalds started using BK around 2002. Much complaining thus fostering new developments in DVCS, if only to be able to get rid of BK.
- After 3 years, Andrew "Tridge" Tridgell started to write a free BK-compatible client. Larry McVoy revoked all free licenses: Linux had no VCS anymore.
- Linus starts looking around, find Monotone too slow and Arch too complicated so…

→ Linus writes his own crude (his words) DVCS: `git`. More a versioning FS than a real DVCS. A community is now growing around `git` (`cogito`, `StGIT`, `gitk`…)

New DVCS appear along with git

Mercurial	Started by M. Mackall a few days after Linus, it is evolving quite fast (despite L. McVoy's efforts to slow it down).
codeville	By Bram Cohen (BitTorrent). Used more for testing merge algorithms, uses BDB for everything.
Bazaar 2	Also known as `bzr`, it is a complete redesign of bazaar, closer to Darcs/Mercurial/BK. Not yet stable. Slow.
svk	A set of Perl scripts above Subversion to add distributed features. Said to be quite stable, bypassing several SVN layers for speed. Correct some Subversion flaws.

And many others... (DCVS, Vesta, etc.)

Agenda

CVS repositories figures

We have 4 different repositories now and *CVSup* can be used to conveniently merge them altogether in a single `/home/ncvs` tree.

Repository	Size (MB)	Directories	Files	csets
doc	183	1653	6171	15594
ports	903	43490	124338	~~138696~~ 150000+
src	1402	9030	60708	~~117233~~ 122238
www	112	595	3479	11836

These include all the history from start of the 2.0 tree back in 1994. Relatively few tags (~~93~~ 97 tags in `Makefile,v`) and branches (~~22~~ 24 right now).

Development process

- 330+ developers, *core team* of 8 and several ad-hoc teams for specific tasks (*portmgr*, *re* and *docmgr* for example)
- Many make their own copy of the trees, work from that and remote commit by just changing `cvs` parameters
- Regular code & feature freezes for releases generate lots of merge requests to `re@`. Trees are sometime locked for weeks...
- `/usr/ports` is never branched to avoid server/mirrors and developers load
- Too many manual intervention on CVS (repo-copies mostly)
- Project-specific branches on Perforce with regular merges (lose specific changes history)

FreeBSD requirements

- Atomic commits & renaming support
- A reference repository (needed for releases and consistency) with *triggers* (mailing of commit messages to lists, tests, . . .)
- Ability to handle big trees such as `/usr/ports` in reasonable times (16 mn to "`cvs co ports`" and a few hours to tag `/usr/src`)
- Automated or mechanically assisted merging
- UI close to CVS
- Efficient storage backend (disk space is not *that* cheap)
- Offline work with full features (commits, tags, and so on)
- Cryptographic signatures of the repository/csets

Agenda

Summary

- Arch is a "protocol" with many implementations (larch, tla, baz)
- Tom Lord has been a pioneer in DVCS for a long time but can be difficult to deal with
- Used to have a big community but it is waning (even though someone has stepped in as maintainer)
- User base has been slow to grow but is there (Debian for example)
- `http://gnuarch.org/`

Architecture

Arch is a cataloging system with a specific namespace:
`archive-name/category--branch--version`

- *Categories* are the main objects that can be branched and versioned (noted `c-b-v`)
- Changesets are created within a specific `c--b--v` tuple and add to the canonical name: `c--b--v-patch-NN`
- An archive is a bag of categories without relation between themselves. Categories can be grouped together with *configs*.
- Categories are the main entity that are branched/committed into
- Changesets are not atomic across configs (see point 1 above)

Problems

The design of Arch/bazaar1 makes for some serious constraints:

- Arch needs to uniquely identify all files
- UI is complicated, making it rather difficult to learn
- Arch is not really space efficient for storage (working trees, revisions libraries/pristine trees, `.arch-cache`)
- Big trees operations are very slow
- Arch/tla just got a new maintainer but seems a dead-end now

Timings for common operations

We now try baz on /usr/src:

Operation	Time	CVS equiv.	Time
`baz import`	11:21	`cvs import`	4:18
`baz get src`	3:28	`cvs co`	14:43
`baz commit -s`	4:29	`cvs commit`	11:52
`baz status`	6:05	`cvs update`	5:22
`baz status`	3:33	`cvs update`	idem

- `cvs import` creates the repository but we need a checkout
- First `status` generates a *revision library* entry
- Timings done on a 2x PIII/800 running 4.11 with SCSI3/ATA disk drives.

Arch study conclusions

- Arch1 is slow, sometimes even slower than CVS
- Arch1 is a low-level VCS that expose too much of its implementation
- Arch1 is mostly dead as an architecture
- Arch2 aka revc is still-born
- Canonical is focused on Bazaar 2 (`bzr`) which still need a lot of work (performance)

Despite all these, it has been fun to work with Arch/baz after 5 years of Perforce. . .

Agenda

Summary

- Started in March, 2005 by M. Mackall after the *BK* incident
- Written in Python with 2 C modules (bdiff, mpatch) for efficiency
- Small code base, very easy installation
- Friendly community & author/contributors around it
- Still evolving (UI, features, add-ons)
- `http://selenic.com/mercurial/`
- April 2006: OpenSolaris has choosen Mercurial as its DVCS

Architecture

- UI close to CVS
- Uses crypto hashes for changesets
- A repository is a branch (more or less)
- Very efficient delta-based storage for metadata
- Very fast even for large trees
- Local or global tags
- Add-ons easy to write and plug ((H)gct, hgk, filters, gpg)
- New storage method: faster and use less inodes (revlogng)

Timings for common operations

We now try Mercurial on `/usr/src`:

Operation	Time	CVS equiv.	Time
`hg clone src`	3:09	`cvs co`	14:43
`hg commit -A`	5:12	`cvs import`	4:18+14:43
`hg commit -m`	0:09	`cvs commit`	5:32
`hg status`	0:06	`cvs update`	3:30

- `clone` and `co` don't do the same exact thing as there is no history in `co` case.
- `cvs import` creates the repository but we need a checkout

Mercurial problems

Mercurial is by far not perfect:

- Binary file support could be better
- Copy/move/renames are not fully implemented in 0.9 (released yesterday)
- ~~Crypto signatures are missing (95% done though)~~
- Limited metadata handling (permissions, ownership)
- Support for partial trees commit/checkout is not yet there
- Documentation is somewhat sparse

Most of these should be fixed for 1.0.

Agenda

Current policies

- Most of our current policies are CVS-oriented
- Worse: some of them are here to workaround around CVS limitations
- Tree freezes before release can last days if not weeks
- Merging is painful and we have to get through `re@`
- Support for older releases is therefore limited
- Perforce "fixes" some issues but has its own

What we need to switch

- Some kind of Patch Queue Manager to manage the official trees (commit mail, branching and so on. . .)
- Teach (evangelise?) developers about the new tool – harder than it seems, it is a touchy subject
- Setting up the Mercurial to CVS mirror
- Conversion of the existing CVS repository?

The Canonical folks already have a PQM, it needs to be adapted to Mercurial

What we would gain from Mercurial

- Easier release management for all
- Easyness and speed of Perforce with distributed features (branching, merging, tags, . . .)
- Full-featured offline work
- Repository replication is natural to Mercurial, no need for CVSup
- Easier setup/support on the `FreeBSD` cluster (no more complicated CVS scripts, no repo-copies, . . .)
- Future: link to the bug tracking system (task-oriented work)
- One VCS for everything. . .

Repository conversion

- It is *really* complicated (different concepts and implementation issues between CVS & Mercurial)
- None of the existing tools can convert everything [yet]. . .
- . . . but we have a promising conversion utility named `cvs2hg`
- It is time-consuming as most tools are slow and we have a *large* number of changesets
- Big issues with repo-copies and manual cvs surgery
- 10k CA$ question: do we need everything in Mercurial?

Agenda

Pros & cons

- A VCS migration is a *lot* of work
- Mercurial is in my opinion up to the task (but not yet feature-complete)
- A proper infrastructure needs to be designed and setup
- New policies & workflows still have to be defined
- Even if FreeBSD doesn't switch, one can use Mercurial for FreeBSD work of course

Thanks!

Thanks to my reviewers: Robert Watson, Mark Murray, Phil Regnauld and Anton Berezin!

And special thanks to Élodie for pushing me into writing this paper.

Slides & paper available on `http://www.keltia.net/` and `http://ns2.freenix.org/~roberto/`

Any question?