

Distributed Source Control

- Introduction
- History
- Current landscape
- Hands-on with hg
- Why hg is fast

Disclaimer

- I'm a dscm newbie
 - But I read a lot

Introduction

- Centralized vs Distributed
 - dscm branches are private until published
 - One branch per bug
- Common dscm features
 - Smart merging
 - Offline use

Introduction, cont.

- Because each checkout is really a separate repository/branch, *you never have to merge **before** checking in your work*
- This eliminates a source of errors, as well as providing a more realistic history of your work
- Eliminates *have vs have not* in open-source projects wrt commit access

History

- 2 generations roughly divided by the Linux kernel moving off of BitKeeper

Generation 1

- Bitkeeper
 - bk
 - Started 1997
 - C++ (?)
- GNU Arch
 - tla, revc
 - Started 2001
 - Shell (!), then C

Generation 1, cont.

- Darcs
 - Started 2002
 - Haskell
- Monotone
 - mtn
 - Started 2003 (?)
 - C++
 - sqlite for storage
- Codeville
 - cdv
 - Started 2004 (?)
 - Python
 - bdb for storage

Generation 2

- Bazaar-NG
 - bZR
 - Jan 05 (before the bk soap opera)
 - Python
- Git
 - C
- Mercurial
 - hg
 - Python

The Arch family tree

- GNU Arch 1.x
 - tla (“Tom Lord's Arch”)
- Bazaar 1
 - Forked in 04
 - baz
- Gnu Arch 2.0
 - Forked in 05 (?)
 - revc
- Bazaar-NG aka Bazaar 2 aka just Bazaar
 - bZR
 - Python rewrite, mostly by baz developers

Competitive landscape in 07

- Inactive
 - tla, revc (Tom Lord: use Bazaar)
 - Codeville
- Still around
 - Monotone
 - Steeper learning curve
 - Slow-ish
- Active
 - hg
 - bZR
 - git
 - darcs

Darcs

- Does not guarantee that old versions will be recoverable (!)
- Exponential scaling issues
 - “When I try to push/pull/apply, darcs just hangs”
- Poor Windows support
- Relatively buggy
 - <http://lists.osuosl.org/pipermail/darcs-users/2007->
- Notable users:
 - Nothing I recognized (but it's de rigeur for the Haskell community)

Git

- Written / used by Linus
- Largest Wikipedia article
- Fast
- Poor windows support
 - Requires cygwin; slow; hates `\r\n`
- Notable users:
 - OLPC
 - Wine
 - X.org

Bazaar

- Supported by Canonical (“the Ubuntu guys”)
- “Foreign branches:” targetting svn (and hg, and git) interop RSN
- Notable users:
 - Drupal
 - Samba
 - Canonical's Launchpad

Mercurial

- Fast
- Excellent documentation
- Trac plugin (beta)
- Notable users:
 - OpenSolaris
 - Xen

Hands on: hg basics

- hg init
- hg add
- hg clone
- hg commit [ci]
- hg status [st]
- hg diff
- hg log

First bit of distributedness

- “svn up” is a 2- part process
 - hg pull
 - Gets changes from the repository it was cloned from
 - hg update [up]
 - Oddly, “update” is aliased to “checkout” and “co” as well (which leads to confusing errors if you write “hg co” when you meant “hg clone”)
 - hg pull -u
 - hg merge
- Whiteboard time!

Demo time

- A simple merge

tags

- Tags really are tags (revision aliases)
 - Not copies
- Use like any other revision marker
 - hg tag foo
 - ...
 - hg up foo

Improved commands

- “revert” can update any file to a specific revision
 - Not just to “unchanged since update” state
- “log -p” includes diff
 - hg diff -r 3867:3868
 - hg log -p 3868

More commands

- annotate
- archive (*export*)
- cat

- incoming
- addremove
- grep (!)
- bisect (extension)
- rollback
- backout

Revisions

- Numbers vs hashes
 - hg identify [id]

hg serve

- Another short demo

Other cool stuff

- Bundles
- Queues

Using hg

- Solo
- Centralized team
- Linus style
- Traditional OSS (pseudo-centralized?)

hg and speed

- Speed affects usability
 - Google's results: if it's slow, you use it less
- Speed enables innovation
 - hg grep
 - queues

Making hg fast

- SCM is mainly IO-bound
 - Seeks are expensive; space is cheap
 - (but, hg is still more space-efficient than svn)
 - Traditional reverse deltas are bad to avoid writes
 - hg's forward deltas
 - Append-only
 - “keyframes” make it $O(1)$
 - Stores stats of files (mtime etc) to avoid reading contents if possible
 - Lock-free design

Making hg fast: python

- struct pack/unpack; string split, join
 - Near-direct access to C
- avoid conditionals in inner loops

My opinion

- Nice for your own code and OSS
- Improved vendor branches
- I miss the Tortoise