The Mercurial SCM

Fast.
Simple.
Distributed.

Bryan O'Sullivan
bos@serpentine.com

About this talk

- Understanding Mercurial
- Working with other people
- Helping you to work efficiently
- Why Mercurial is fast
- Why your tools are important
- Where Mercurial is going

My assumptions about you

- You want to manage some files
  - Source code
  - Web site
  - Home directory
  - Great American Novel
- You're not afraid of the shell or command prompt

About Mercurial

- Work began in April 2005
  - Goal: manage Linux-sized trees efficiently
- Now 95% pure Python, 750 lines of C for speed
- Rapid uptake due to speed and usability:
  - **Linux**: video4linux, ALSA, e2fsprogs, ...
  - **System software**: Xen, OpenSolaris, Conary, FreeBSD ports, ...
  - **Other exciting projects**: One Laptop Per Child, Moin Moin, microformats, physics textbooks, ...

Revision control: a crowded field

Accurev

BitKeeper *

CCC/Harvest

ClearCase *

Perforce *

StarTeam *

Surround

Vault

...

Aegis

Arch

Bazaar-NG *

CVS *

Darcs *

git *

PRCS

Subversion *

SVK

Vesta

...

A developer's POV

- I have work to do!
- I want something *simple* that *works*
- My SCM tool should:
  1. Be easy to understand
  2. Help me to work with others
  3. Let me work efficiently

A developer's POV

- My SCM tool should:
  1. ***Be easy to understand***
  2. Help me to work with others
  3. Let me work efficiently

Be easy to understand

- **User quote:** “Mercurial's conceptual model is clean and simple enough to carry around in my head”

- Let's introduce three concepts:
  - Repository
  - Changeset
  - Working directory

What's a repository?

- Simple
  - A directory containing the history of my project
  - No fancy database, no big server: just a directory
- Lightweight
  - Making a copy (a “clone”) of a repository is cheap
- Everywhere
  - All work happens in repositories
  - Every person works in their own repositories

What's *in* a repository?

Mercurial doesn't actually expose these details.

(But they're simple, and it helps to know what's going on.)

- Changelog
  - The history of changes to the repository
- Manifest
  - History of file versions used in each changeset
- Per-file data
  - History of every file that Mercurial tracks

Contrast the repository models

	Traditional SCM	**Mercurial**
Central repo	Exactly one	As many as needed
Bottlenecks	Central server	None
Load mgmt	Expensive or impossible	Mirrors wherever, for free
Distant users	Slow server response	Fast local response
Server failure	Catastrophic	Full backup in every repo
Network connection	Always needed	Fully productive anywhere

What's a changeset?

- A snapshot of the project at a point in time
- It records:
  - Who made the change (the “committer”)
  - A readable description of the change
  - What files were changed, what the changes were
  - What the parent changeset was
- Creating a changeset is called “committing”i t

What's the working directory?

- A view of the repo as of some changeset
  - This changeset is the working directory's *parent*
- I can edit any file in the working directory
  - My changes will get rolled into the next changeset
  - I can add, remove, rename, and copy files
- I can see what I've changed, and how

Micro-tutorial: Hg in 60 seconds

1. Create a repository — `hg init myrepo`
2. Go in there — `cd myrepo`
3. Edit a file — `emacs myfile`
4. Tell hg to track the file — `hg add myfile`
5. Now what's happening? — `hg status`
   - “File has been added” — *`A myfile`*
6. Record my changes — `hg commit`

A developer's POV

- My SCM tool should:
  1. Be easy to understand
  2. ***Help me to work with others***
  3. Let me work efficiently

Parallel play

- People naturally work in parallel
  - Most revision control tools make this *hard*
- I make some changes
- I go to check them in ("commit" them)
- What if someone else committed first?

I have commitment issues

- What if someone else committed before me?
- I have to merge their changes *before* I can commit
- **There's no permanent record of my changes yet!**
- A mistake during the merge can **lose my work**

The Hg model: branching

- Remember that a changeset has a parent?
- Two changesets with the same parent make a *branch*
- That's *all* a branch is!
  - Nothing dramatic or complex

The Hg model: merging

- What do we do with branches?
- Some changesets have *two* parents
- These are *merge* changesets
- A merge changeset simply says:
  - "This is how I merged changesets A and B"

Plays well with others

- Mercurial *naturally supports* parallel work
- I commit my changes when it suits me
- My changes are clean and self-contained
- I don't merge with your changes until *after* I've committed mine

Merging without stress

- What if I make a mistake during a merge?
- My changes are still there; so are yours
- **No work gets lost**
- I simply redo the merge

Sharing is easy

- Built-in web server
  - CGI server for Apache integration
- Use ssh (“secure shell”) for secure remote access
- Works over network filesystems
- Share work offline using email, USB flash, ...

Sharing is symmetric

- I **clone** a remote repo to get a local copy
- I **pull** new changes from a remote repo
- I **push** my changes to another repo
- After a push, the remote repo is identical to mine

Sharing mini-tutorial

hg commit *I'm finished with my changes*

hg push *Push my changes to another repo*

abort: unsynced remote changes! *Hg tells me I need to merge first*

hg pull *Pull remote changes into my repo*

added 1 changeset with 1 change to 1 file

(run 'hg heads' to see heads, 'hg merge' to merge)

hg merge *Merge their changes with mine*

hg commit *Commit the result of the merge*

hg push *Push my changes and the merge*

added 2 changesets with 2 changes to 2 files

A developer's POV

- My SCM tool should:
  1. Be easy to understand
  2. Help me to work with others
  3. ***Let me work efficiently***

Mercurial is very, very fast

- File architecture eliminates seeks
- Common operations are cheap
  - No need to wait for the software
- All data is local
  - No need to wait for the network
- Metadata is shared between local repos

Size matters: examining Linux

- A Linux kernel source tree is 262 MB in size
  - Just the files from a tarball, no history of any kind
- 1 year of Linux kernel history in Mercurial
  - 19,500 files tracked
  - 27,500 changesets committed (1,840 merges)
  - ~1,000 contributors
- Complete history: 293 MB

Some performance numbers

- Numbers from my 3-year-old laptop
- Used Linux kernel repo

Frequent tasks	**Time (secs)**
Find changes in working directory	2.2
Commit change to one file	1.6
Annotate 3200-line file with 500-rev history	3.5
Make local clone	2.3
Less common tasks: still fast!	
Populate working directory after clone	70.0
Pull 350 changesets over http	30.0

Mercurial makes me more efficient

- Simple concepts let me focus
  - Think less about SCM, more about work
- Commits and merges are separate
  - Harder to lose or corrupt work by accident
- Cheap repos let me sandbox my work
  - One repo per task
- Local repos let me work anywhere
  - On train, over slow net connection, you name it

Why is Mercurial fast?

- All I/O is optimised
  - Simple file formats
  - Avoid disk I/O
  - Prefer “s treaming” I/O

Simple file formats

- Only two on-disk data structures
  - Revlog (file metadata, manifest, changelog)
  - Dirstate (working dir)
  - Less variety allows focus on speed
- File formats efficiently parsed in pure Python
  - Few Python conditionals: “if” means “slow”
  - Much use of struct.pack, string.split

Avoid disk I/O

- Metadata files are indexed
- Small files: index & data combined
  - Reduces internal file fragmentation & seeks
  - Gives 40% space reduction
- Dirstate: fast status without open/read
  - Stores size, mtime, mode of each file
  - Only os.stat needed for most status checks

Prefer “streaming” I/O

- Files accessed in alphabetical order
  - Many filesystems optimise for this pattern
- Metadata laid out for linear reads
  - Forward deltas with periodic fulltext
- Need to read backwards (e.g. “log”)?
- Be clever!
  - Read forwards in *chunks*, then reverse output
  - ~1000x faster than reading backwards

Other useful things to know about

- Web support includes history browser
- Comprehensive hook support
  - Automate important actions (builds, code checks)
- Add optional features using extension modules
  - GUI tools
  - Patch queue management
  - Automated regression search

Why your choice of tools matters

- Your tools shape how *you* work
  - RCS/SCCS: I have to be logged into a host to work
  - CVS/SVN: I have to be online to work
  - Distributed tools: I can work **anywhere, any time**
- Your tools shape how *others* work with you
  - RCS/SCCS: outsiders can't see history
  - CVS/SVN: outsiders can read, but not write
  - Distributed tools: there are **no outsiders**

Common FUD

- “I'm afraid my project will fork”
  - I can already suck out your history from CVS & SVN, so it's a **lost cause**
  - Forking is fundamentally a **social issue**
  - Distributed tools make **reconciliation** after a fork easy
- “I want something simple”
  - The best distributed tools are **simpler** than the central ones
  - Prefer a central model? Simply use the tool that way

Distributed tools and Free Software

- **Fundamentally and absolutely better** than centralised tools
- **Eliminate** the “have” / “have not” distinction
- **Choose** the development model you like best
- **Every user** becomes a potential contributor
- Good branching and merging let you **scale** to match your success

Coming attractions*

- In progress:
  - Support for merging changes across renames
  - Comprehensive user manual
  - Eclipse support
- Want to add:
  - Better GUI interfaces on Windows, Linux, MacOS
  - Integration with other popular IDEs

* Based on April 2006 User Survey

User survey quotes: the team

- “The developers are super-helpful.”
- “The Mercurial community is more polite and helpful than most.”
- **“The community is great around Mercurial.”**

User survey quotes: the software

- “I was up and ready to go with Mercurial in less than 5 minutes.”
- “Mercurial was extraordinarily easy to learn.”
- “I used to like CVS a lot. I can't imagine going back. Really.”
- **“I consider Mercurial the best version control system on earth.”**

Thank you!