Distributed versioning for everyone

Nicolas Pouillard

Nicolas.Pouillard@inria.fr

March 20, 2008
Outline

1. Introduction
2. Principles of Distributed Versioning
3. Darcs is one of them
4. Conclusion
SCM: “Source Code Manager”

- Keeps track of changes to source code so you can track down bugs and work collaboratively.
- Most famous example: CVS
- Numerous acronyms: RCS, SCM, VCS
- DSCM: Distributed Source Code Manager
Purpose

What’s the purpose of this presentation

- Show the importance of the distributed feature
- Enrich your toolbox with a DSCM
- Exorcize rumors about darcs
- Show how DSCM are adapted for personal use

What’s not the purpose of it

- A flame against other DSCMs
- A precise darcs tutorial
- A real explanation of the Theory of patches
Outline

1. Introduction
2. Principles of Distributed Versioning
3. Darcs is one of them
4. Conclusion
Distributed rather than centralized

Centralized

Examples: CVS, Subversion, Perforce

Distributed

Examples: darcs, Git, Bitkeeper, monotone, arch
Principles

Unify Repositories and Working copies
- Working copies with full history
- Repositories with local changes

Users record/commit in a local branch
- Local branches can be then merged with remote ones
- Branching/Merging is then **forced** to work
Local branches

Branching in a centralized system is morally flawed
- People use branches only when they must
- Branches are public (not discreet)
- That’s considered as an advanced usage

Distributed systems make them easy
- Offline commit (no need to be connected)
- Try out an idea (cheap and discreet)
- Polish your work / amend a patch
- Publish with a delay (e.g. end of the work-day)
DSCM are often lighter

- No server to setup
- Make a repository is as easy as "darcs initialize"
- There is no need to "wait for" a center
- No commit rights management needed
Collaborating to an open source project

- Local branches is a "must have"
- Help to publish only clean and working changes
- Send your patches under your name

Work with user contributions

- Maintain an auto-gratification principle
- No need for commit rights (was really a pain)
- Commutation is essential
- Delaying user contributions if needed
Outline

1. Introduction

2. Principles of Distributed Versioning

3. Darcs is one of them
   - Darcs overview
   - Darcs Theory of patches
   - Darcs for working alone
   - Darcs branching, merging, tagging
   - Darcs for working with others
   - Conflicts and concerns

4. Conclusion
Outline

1. Introduction

2. Principles of Distributed Versioning

3. Darcs is one of them
   - Darcs overview
     - Darcs Theory of patches
     - Darcs for working alone
     - Darcs branching, merging, tagging
     - Darcs for working with others
     - Conflicts and concerns

4. Conclusion
Why focusing on one of them

- Treat each system in details is too long
- Abstract over DSCM would be too obscure
- Darcs is conceptually simple
- Darcs is certainly the smarter of them
Ideas behind darcs

Distributed
- A simple “egalitarian” distributed model
- “Cherry picking” of changes
- Avoidance of “merge points” (no merge history)

Interactive
- Efficient and easy to learn
- Improved work flow (e.g. partial records, code review, ...)

Smart
- Based on a unique algebra of patches
- Spontaneous branches
- Commutation of changes
Distributed versioning for everyone
Darcs is one of them
Darcs overview

Change-based rather than version-based

Version-based

Examples: Git, Bitkeeper, Monotone, CVS, Subversion

Change-based

Examples: darcs
Darcs terminology

- A change is a logical entity
- A patch is a description of a change
- The state of a repository is defined by its set of changes
- A set of changes is stored as a sequence of patches

Notation

- A change is represented as a capital letter: $A$
- A patch is represented by a capital letter with possibly primes and/or a subscript: $A, A', A_1$
- Sometimes the state (or context) before and after a patch is represented by lowercase superscripts: $oA^a$
The state of a repository is defined by a set of changes.
The repository is represented by a sequence of patches.
Outline

1. Introduction

2. Principles of Distributed Versioning

3. Darcs is one of them
   - Darcs overview
   - Darcs Theory of patches
   - Darcs for working alone
   - Darcs branching, merging, tagging
   - Darcs for working with others
   - Conflicts and concerns

4. Conclusion
Some primitive patches

- **addfile** $f$: Create the empty file $f$
- **rmdir** $d$: Remove the empty directory $d$
- **move** $x$ $y$: Move/rename the file/directory $x$ into $y$
- **hunk**: Change the contents of a file
  
  hunk "foo.txt" 42
  - the old lines has been
  - removed.
  + and replaced by this one

However the theory is independent of its primitives
Each patch is invertible

Definition

$$\text{invert} \circ A^a = a A^{-1}$$

Property

$$\forall x. \text{invert} (\text{invert} x) = x$$

Examples

- $$\text{invert} (\text{addfile } f) = \text{rmfile } f$$
- $$\text{invert} (\text{move } x y) = \text{move } y x$$
- $$\text{invert} (\text{hunk } f \text{ line old new}) = \text{hunk } f \text{ line new old}$$
- $$\text{invert} (A :> B) = (\text{invert } B) :> (\text{invert } A)$$

Consequence: While move is easy, copy hardly make sense
Independent changes $\Rightarrow$ commuting patches

\[ oA^aB^b \leftrightarrow oB^c_1A^b_1 \]

Examples
- Hunks on different files trivially commute
- Hunks commute with moves
- Hunks on different parts of a file commute (output patches have different line numbers)
Illustrated naive merging...
Distributed versioning for everyone
Darcs is one of them
Darcs Theory of patches

Nicolas Pouillard
March 20, 2008 24 / 48
Outline

1 Introduction

2 Principles of Distributed Versioning

3 Darcs is one of them
   - Darcs overview
   - Darcs Theory of patches
   - Darcs for working alone
   - Darcs branching, merging, tagging
   - Darcs for working with others
   - Conflicts and concerns

4 Conclusion
Darcs for small projects

- A research paper
- A prototype implementation
- A small module/library
- Configuration files
- Personal web page
- More to imagine...
Distributed versioning for everyone
Darcs is one of them
Darcs for working alone

The bare minimum

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialize</td>
<td>Initialize a new source tree as a darcs repository</td>
</tr>
<tr>
<td>add</td>
<td>Add one or more new files or directories</td>
</tr>
<tr>
<td>record</td>
<td>Save local changes as a patch</td>
</tr>
<tr>
<td>mv</td>
<td>Move/rename one or more files or directories</td>
</tr>
<tr>
<td>whatsnew</td>
<td>Display local/unrecorded changes</td>
</tr>
</tbody>
</table>
Handy operations

- **revert**: Revert to the recorded version (not always doable)
- **unrevert**: Undo last revert (unless changes after the revert)
- **rollback**: Record a new patch reversing some changes

`revert ; unrevert ≈ id`
### Overriding unpublished changes

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>amend-record</td>
<td>Replace a patch with a better version</td>
</tr>
<tr>
<td>unrecord</td>
<td>Remove patches wo/ changing the working copy</td>
</tr>
<tr>
<td>obliterate</td>
<td>Delete selected patches from the repository</td>
</tr>
</tbody>
</table>

- amend-record ≈ unrecord ; record
- obliterate ≈ unrecord ; revert
Towards advanced patch types

**replace**  Replace a token with a new value for that token
Looking in the past

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>changes</code></td>
<td>Give a summary of the repository history</td>
</tr>
<tr>
<td><code>annotate</code></td>
<td>Display which patch last modified something</td>
</tr>
<tr>
<td><code>diff</code></td>
<td>Create a diff between versions of the repository</td>
</tr>
<tr>
<td><code>dist</code></td>
<td>Create a distribution tarball</td>
</tr>
<tr>
<td><code>trackdown</code></td>
<td>Locate the most recent version lacking an error</td>
</tr>
<tr>
<td><code>show</code></td>
<td>Show information which is stored by darcs</td>
</tr>
</tbody>
</table>
Outline

1. Introduction

2. Principles of Distributed Versioning

3. Darcs is one of them
   - Darcs overview
   - Darcs Theory of patches
   - Darcs for working alone
   - Darcs branching, merging, tagging
   - Darcs for working with others
   - Conflicts and concerns

4. Conclusion
Moving patches around

pull  Apply patches from another repository to this one
push  Apply patches from this repository to another one
get   Create a local copy of another repository
put   Make a copy of the repository
send  Send (by email) a bundle of one or more patches
apply Apply patches (from an email) to the repository

- get src dst ≈ initialize src ; cd src && pull dst
- put dst ≈ initialize dst ; push dst
- push dst ≈ send dst ; cd dst && apply
Branching and merging

Branching is as easy as copying all patches

$ darcs get foo-stable foo-with-feature-A

Merging is as easy as \{pull, push\}ing things

$ cd foo-with-feature-A
$ darcs pull ../foo-stable
Tagging your repository (darcs tag)

A ”tag” patch is:
- A change with no effect
- Transitively depends on all patches
- Really depends only on non tagged patches

Tagging quite often is a good practice
- Tag (some/only) versions that pass all tests
- Tag pre-releases and releases

Drawback of tagging
Freeze commutations (patches under a tag cannot cross the tag)
Distributed versioning for everyone
Darcs is one of them
Darcs branching, merging, tagging

darcs + tagging like crazy \(\approx\) git

\[
\text{
$\text{git commit} = \text{darcs record} \; ; \; \text{darcs tag -m } <\text{SHA1}>$
}
\]

- Enforce the history
- Enforce the order of patches
- Loose commutativity (manually hacked with git rebase)
Fancy features

- Take the union: pull another repository
- Extract a sub part: pull interactively only what’s needed
Outline

1 Introduction

2 Principles of Distributed Versioning

3 Darcs is one of them
   - Darcs overview
   - Darcs Theory of patches
   - Darcs for working alone
   - Darcs branching, merging, tagging
   - Darcs for working with others
   - Conflicts and concerns

4 Conclusion
Working with others

- Just replace pathnames by URLs (http, ssh)
- Use send/apply for email based contributions
Outline

1. Introduction

2. Principles of Distributed Versioning

3. Darcs is one of them
   - Darcs overview
   - Darcs Theory of patches
   - Darcs for working alone
   - Darcs branching, merging, tagging
   - Darcs for working with others
   - Conflicts and concerns

4. Conclusion
When conflicts happen

Two patches conflicts

- they are parallel patches ($A \lor B$)
- they don’t commute ($A B^{-1} \Leftrightarrow B' A'^{-1}$)

Conflict example

hunk "foo.txt" 42
- # TODO
+ # FIXED

hunk "foo.txt" 42
- # TODO
+ # DONE
Resolving conflicts

- When two patches conflict one add a third one
- By depending on the conflicting patches it tells what to do
- Resolutions patches should be shared as much as possible
Avoiding conflicts

Recipe

- Pull often
- Amend local patches to resolve conflicts
- Push/send clean patches
Is darcs slow?

- Performances are due to its algorithms not its implementation
- Darcs algorithms provide more power/flexibility
- Completely usable for day to day commands
- Can be really slow on hard requests
- Darcs2 has made great progress
- Darcs2 reports progress to the user
- Darcs2 handles the conflict resolution problem
Are DSCCM slow or greedy?

- Full history means bigger/slower copies/gets
- Hard links in the repository
- More network friendly than CVS/SVN
- Darcs2 partial repositories could help
Outline

1. Introduction

2. Principles of Distributed Versioning

3. Darcs is one of them

4. Conclusion
Conclusion and questions

So, convinced?
Resources

- ”The Monad Reader”, issue 9 by Jason Dagit
- ”Implementing the darcs patch formalism ...and verifying it” by David Roundy
- The darcs website http://darcs.net
- The darcs help