

Fast, Flexible and Fun: Revision Control with Mercurial

Martin Geisler
<mg@aragost.com>

Mercurial Geek Night
January 13th, 2011



About the Speaker

Martin Geisler:

- ▶ core Mercurial developer:
 - ▶ reviews patches from the community
 - ▶ helps users in our IRC channel
- ▶ PhD in Computer Science from Aarhus University, DK
 - ▶ exchange student at ETH Zurich in 2005
 - ▶ visited IBM Zurich Research Lab in 2008
- ▶ now working at aragost Trifork, Zurich
 - ▶ offers professional Mercurial support
 - ▶ customization, migration, training
 - ▶ advice on best practices



Outline

Mercurial Introduction

- Overview

- Centralized vs Distributed

- Key Mercurial Concepts

Using Mercurial

- Workflows

- Branches

Extensions and Frontends

Wrapping Up



Outline

Mercurial Introduction

Overview

Centralized vs Distributed

Key Mercurial Concepts

Using Mercurial

Workflows

Branches

Extensions and Frontends

Wrapping Up



Outline

Mercurial Introduction

Overview

Centralized vs Distributed

Key Mercurial Concepts

Using Mercurial

Workflows

Branches

Extensions and Frontends

Wrapping Up



What is Mercurial?

Main features:

- ▶ fast, **distributed** revision control system
- ▶ robust support for branching **and** merging
- ▶ very flexible and extensible



Who is Using it?

Mercurial is used by:

- ▶ Oracle for Java, OpenSolaris, NetBeans, OpenOffice, ...
- ▶ Mozilla for Firefox, Thunderbird, ...
- ▶ Google
- ▶ many more...



OpenJDK



Outline

Mercurial Introduction

Overview

Centralized vs Distributed

Key Mercurial Concepts

Using Mercurial

Workflows

Branches

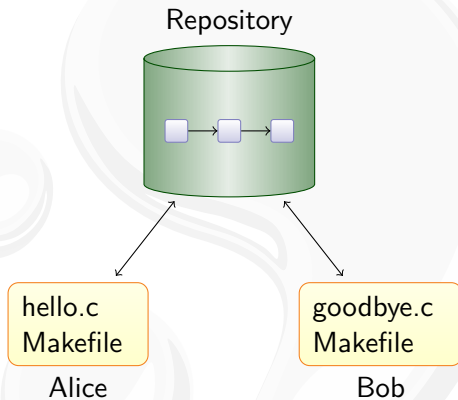
Extensions and Frontends

Wrapping Up



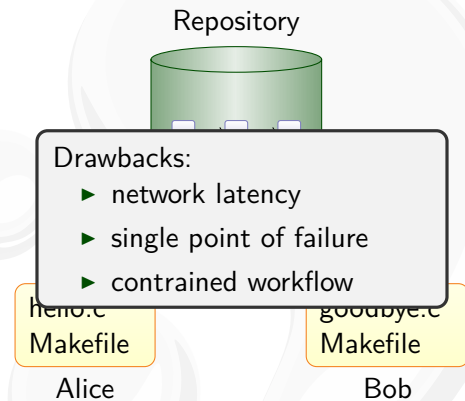
Centralized Revision Control

Single repository, multiple working copies:



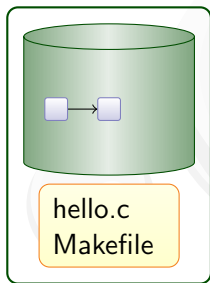
Centralized Revision Control

Single repository, multiple working copies:

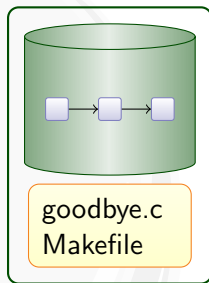


Distributed Revision Control

Mercurial duplicates the history on many servers:



Alice

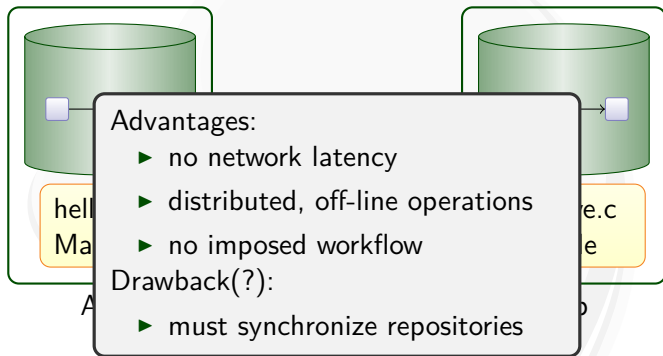


Bob



Distributed Revision Control

Mercurial duplicates the history on many servers:



Outline

Mercurial Introduction

Overview

Centralized vs Distributed

Key Mercurial Concepts

Using Mercurial

Workflows

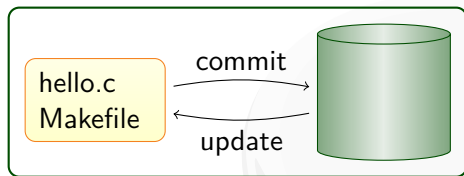
Branches

Extensions and Frontends

Wrapping Up



Key Mercurial Commands



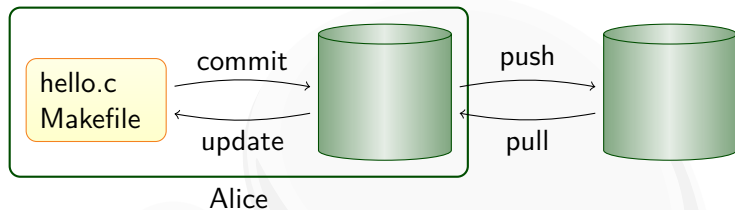
Alice

Local commands:

- ▶ `hg commit`: save a snapshot into the current repository
- ▶ `hg update`: checkout revision into working directory
- ▶ `hg merge`: join different lines of history



Key Mercurial Commands



Local commands:

- ▶ `hg commit`: save a snapshot into the current repository
- ▶ `hg update`: checkout revision into working directory
- ▶ `hg merge`: join different lines of history

Network commands:

- ▶ `hg pull`: retrieve changesets from another repository
- ▶ `hg push`: send your changesets to another repository

Moving Changesets Around

Pull and merge:

Alice



Bob



Moving Changesets Around

Pull and merge:

Alice



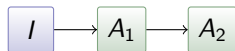
Bob



Moving Changesets Around

Pull and merge:

Alice



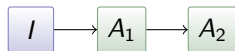
Bob



Moving Changesets Around

Pull and merge:

Alice

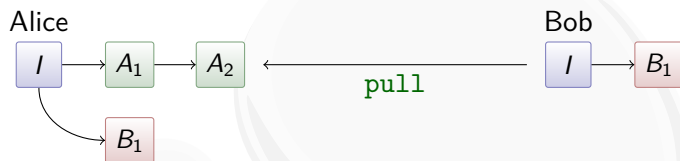


Bob



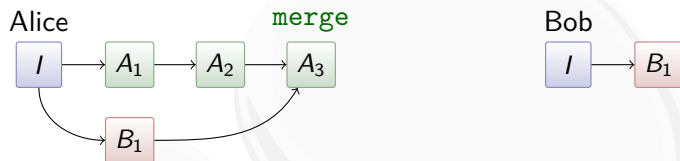
Moving Changesets Around

Pull and merge:



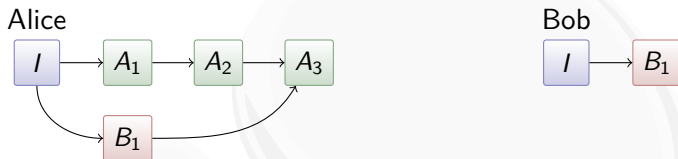
Moving Changesets Around

Pull and merge:



Moving Changesets Around

Pull and merge:



Merging:

- ▶ find common ancestor of A_2 and B_1 : I
- ▶ compute differences between I and B_1
- ▶ apply them to A_2 , taking renames into account



Outline

Mercurial Introduction

Overview

Centralized vs Distributed

Key Mercurial Concepts

Using Mercurial

Workflows

Branches

Extensions and Frontends

Wrapping Up



Outline

Mercurial Introduction

Overview

Centralized vs Distributed

Key Mercurial Concepts

Using Mercurial Workflows

Branches

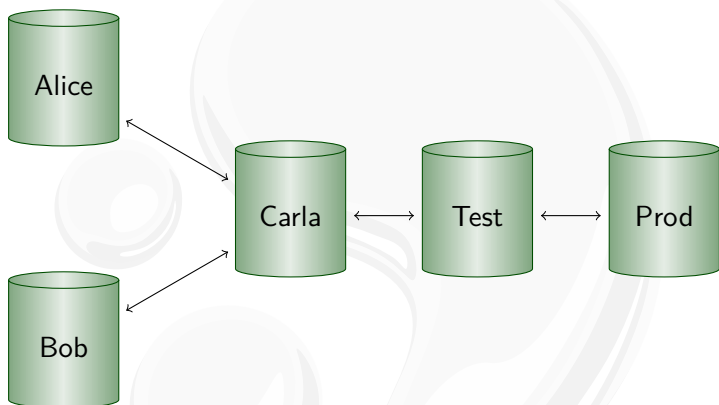
Extensions and Frontends

Wrapping Up



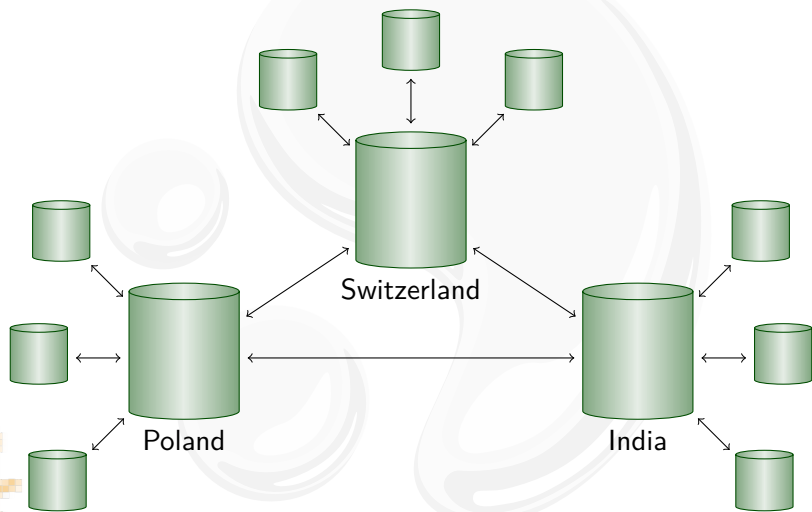
Workflow in a Team

Mercurial scales from a single team. . . :



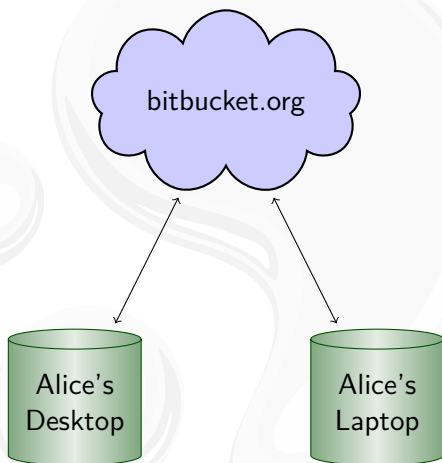
Workflow Between Company Divisions

... to enterprise-wide development. ... :



Workflow Between Two Computers

... to working with yourself:



Outline

Mercurial Introduction

Overview

Centralized vs Distributed

Key Mercurial Concepts

Using Mercurial

Workflows

Branches

Extensions and Frontends

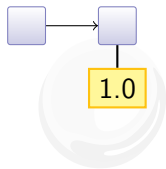
Wrapping Up



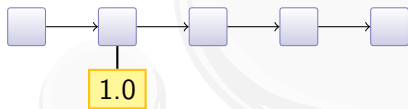
Release Branches



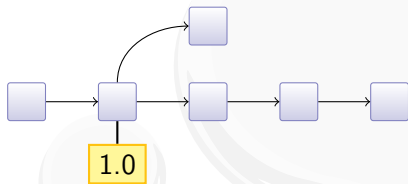
Release Branches



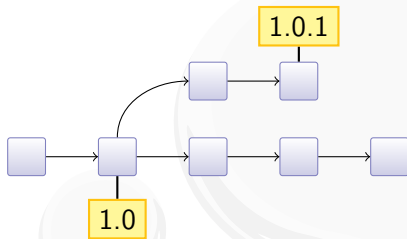
Release Branches



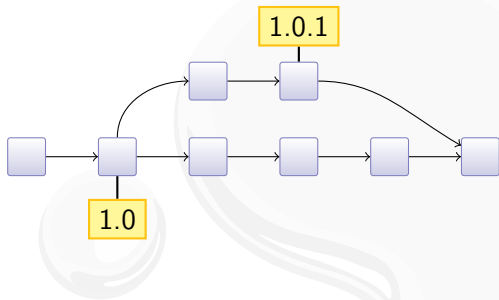
Release Branches



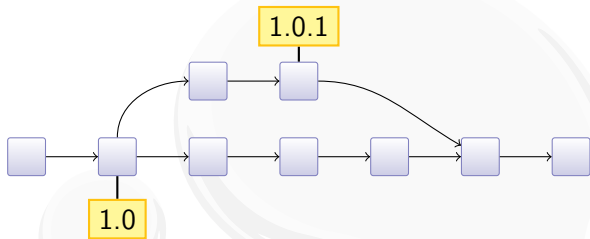
Release Branches



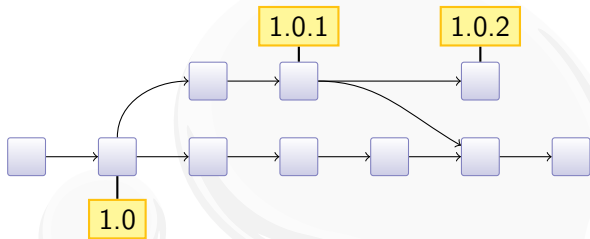
Release Branches



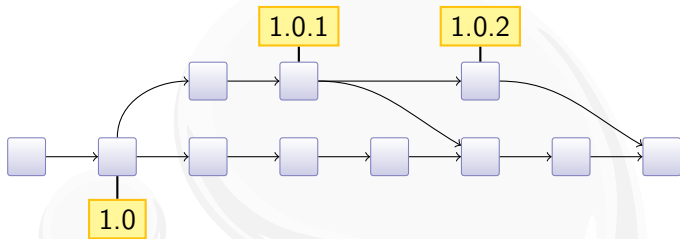
Release Branches



Release Branches



Release Branches



Outline

Mercurial Introduction

- Overview

- Centralized vs Distributed

- Key Mercurial Concepts

Using Mercurial

- Workflows

- Branches

Extensions and Frontends

Wrapping Up



Mercurial is Extensible

You can add new functionality to Mercurial:

- ▶ ships with 30+ extensions
- ▶ wiki lists 75+ extensions
- ▶ extensions can change basically everything
- ▶ helps to keep the core small and focused



Moving Changesets Around

Tired of all those merges? Use the **rebase** extension!

- ▶ Revision graph:



Moving Changesets Around

Tired of all those merges? Use the **rebase** extension!

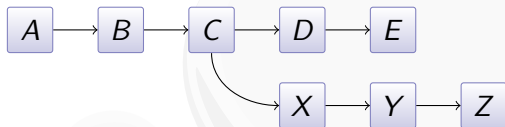
- ▶ Revision graph:



Moving Changesets Around

Tired of all those merges? Use the **rebase** extension!

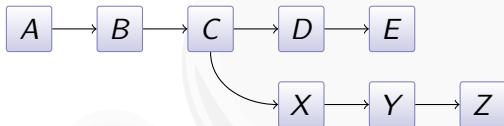
- Revision graph:



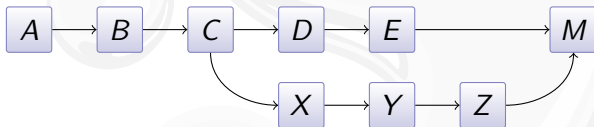
Moving Changesets Around

Tired of all those merges? Use the **rebase** extension!

- ▶ Revision graph:



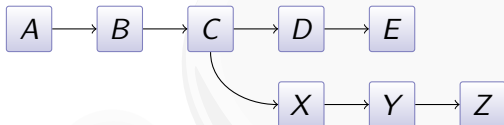
- ▶ Merge:



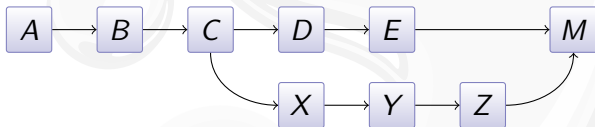
Moving Changesets Around

Tired of all those merges? Use the **rebase** extension!

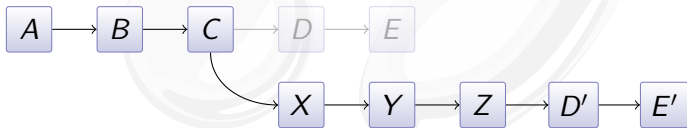
- ▶ Revision graph:



- ▶ Merge:



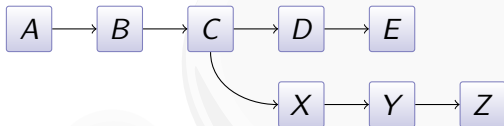
- ▶ Rebase:



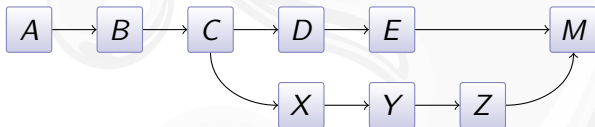
Moving Changesets Around

Tired of all those merges? Use the **rebase** extension!

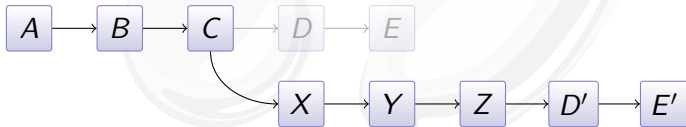
- ▶ Revision graph:



- ▶ Merge:



- ▶ Rebase:



- ▶ Beware: public changes should never be rebased.



Interfacing with Subversion

The `hgsubversion` extension let's you:

- ▶ use `hg clone` on a SVN URL
- ▶ use `hg pull` to convert new SVN revisions
- ▶ use `hg push` to commit changesets to SVN server

Goal: make `hg` a better Subversion client than `svn`!



Third-Party Tools

Tools with Mercurial support:

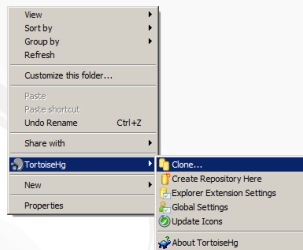
- ▶ Graphical frontends: TortoiseHg, MacHg, ...
- ▶ IDEs: Eclipse, NetBeans, IntelliJ, Visual Studio, ...
- ▶ Project Support: Trac, JIRA, Maven, Hudson, ...



TortoiseHg

Excellent graphical frontend for Mercurial:

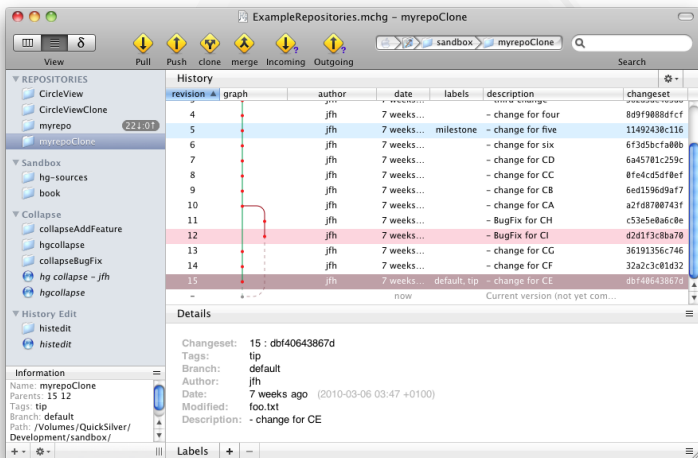
- ▶ works on Windows, Mac, Linux
 - ▶ complete Mercurial installation for Windows
 - ▶ Windows Explorer integration (right-click menu)



- ▶ support for popular extensions:
 - ▶ **mq** for managing patches
 - ▶ **hgsubversion** for interfacing with SVN



Fast, native Mercurial frontend for Mac OS X:



The screenshot shows the MacHg application window titled "ExampleRepositories.mchg - myrepoClone". The interface includes a menu bar with "View", "Pull", "Push", "clone", "merge", "Incoming", and "Outgoing". A sidebar on the left lists repositories and a "Collapse" section. The main area displays a "History" table with columns for revision, graph, author, date, labels, description, and changeset. Below the history is a "Details" section for the selected revision 15.

revision	graph	author	date	labels	description	changeset
4		jfh	7 weeks ago		- change for four	8d9f908dfcf
5		jfh	7 weeks ago	milestone	- change for five	11492430c116
6		jfh	7 weeks ago		- change for six	6f3d5bcfa00b
7		jfh	7 weeks ago		- change for CD	6a45701c259c
8		jfh	7 weeks ago		- change for CC	0fe4cd5df0ef
9		jfh	7 weeks ago		- change for CA	6ed1596d9af7
10		jfh	7 weeks ago		- change for CB	a2fd8700743f
11		jfh	7 weeks ago		- BugFix for CH	c535e8a6c0e
12		jfh	7 weeks ago		- BugFix for CI	d2d1f3c8ba70
13		jfh	7 weeks ago		- change for CG	36191356c746
14		jfh	7 weeks ago		- change for CF	32a2c3c01d32
15		jfh	7 weeks ago	default, tip	- change for CE	dbf40643867d
-			now		Current version (not yet com...	

Details

Changeset: 15 : dbf40643867d
Tags: tip
Branch: default
Author: jfh
Date: 7 weeks ago (2010-03-06 03:47 +0100)
Modified: foo.txt
Description: - change for CE

Outline

Mercurial Introduction

Overview

Centralized vs Distributed

Key Mercurial Concepts

Using Mercurial

Workflows

Branches

Extensions and Frontends

Wrapping Up



Mercurial in a Nutshell

Mercurial changes the way you develop:

- ▶ simple yet strong model for **both** branching and merging
- ▶ power tool instead of necessary evil
- ▶ light-weight and snappy



More Information

- ▶ Mercurial homepage:
<http://mercurial.selenic.com/>
- ▶ *Mercurial: The Definitive Guide*:
<http://hgbook.red-bean.com/>
- ▶ Getting Started:
<http://mercurial.aragost.com/kick-start/>
<http://mercurial.ch/>
<http://hginit.com/>
- ▶ Some free Mercurial hosting sites:
<http://bitbucket.org/>
<http://code.google.com/>
<http://sourceforge.net/>
<http://www.codeplex.com/>



Contact

Please get in touch if you have more questions or have considered using Mercurial in your organization:

- ▶ Email: mg@aragost.com
- ▶ IRC: [mg](#) in [#mercurial](#) on [irc.freenode.net](#)



Mercurial Contributors

From <http://ohloh.net/p/mercurial/map:>



Showing 50 of [325 contributors](#).



Mercurial Contributors

From <http://ohloh.net/p/mercurial/map:>



Showing 50 of [325 contributors](#).



Outline

Performance Study: OpenOffice

Subversion and Branches

The Underlying Model

Using History



OpenOffice

Fairly large repository:

- ▶ 70,000 files, 2,0 GB of data
- ▶ 270,000 changesets, 2,3 GB of history



Mercurial is still fast on a repository of this size:

```
$ time hg status
0.45s user 0.15s system 99% cpu 0.605 total
$ time hg tip
0.28s user 0.03s system 99% cpu 0.309 total
$ time hg log -r DEV300_m50
0.30s user 0.04s system 99% cpu 0.334 total
$ time hg diff
0.74s user 0.16s system 88% cpu 1.006 total
$ time hg commit -m 'Small change'
1.77s user 0.25s system 98% cpu 2.053 total
```

Outline

Performance Study: OpenOffice

Subversion and Branches

The Underlying Model

Using History



Branches in SVN

Subversion knows nothing about branches!

- ▶ but SVN has a cheap copy mechanism
- ▶ used for tags and branches



Branches in SVN

Subversion knows nothing about branches!

- ▶ but SVN has a cheap copy mechanism
- ▶ used for tags and branches

r10

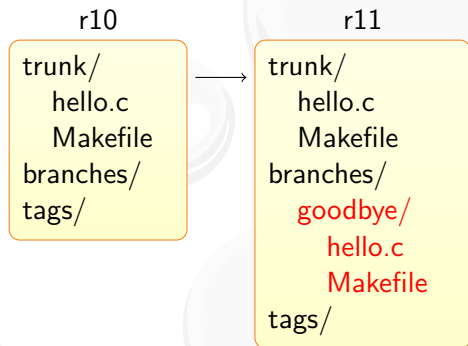
```
trunk/  
  hello.c  
  Makefile  
branches/  
tags/
```



Branches in SVN

Subversion knows nothing about branches!

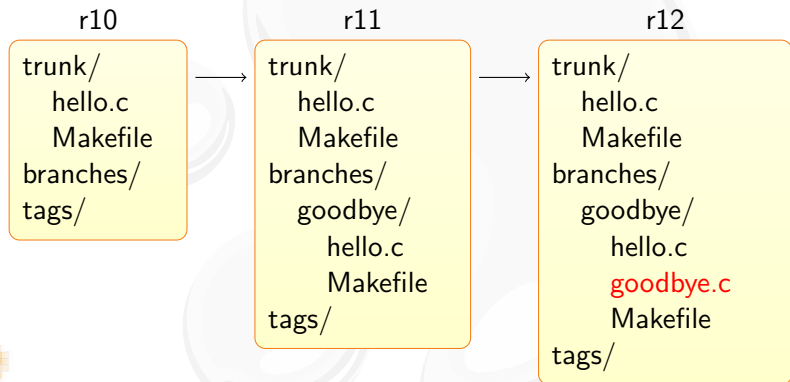
- ▶ but SVN has a cheap copy mechanism
- ▶ used for tags and branches



Branches in SVN

Subversion knows nothing about branches!

- ▶ but SVN has a cheap copy mechanism
- ▶ used for tags and branches



Merging Branches in SVN

The support is incomplete and fragile:

- ▶ renamed files are not merged correctly
- ▶ old clients will not update the merge info



Merging Branches in SVN

The support is incomplete and fragile:

- ▶ renamed files are not merged correctly
- ▶ old clients will not update the merge info

From the SVN Book:

The bottom line is that Subversion's merge-tracking feature has an **extremely complex** internal implementation, and the `svn:mergeinfo` property is the only window the user has into the machinery. Because the feature is **relatively new**, a numbers of edge cases and possible unexpected behaviors may pop up. —*Version Control with Subversion*

(Mercurial has robust built-in support for merging branches.)



Outline

Performance Study: OpenOffice

Subversion and Branches

The Underlying Model

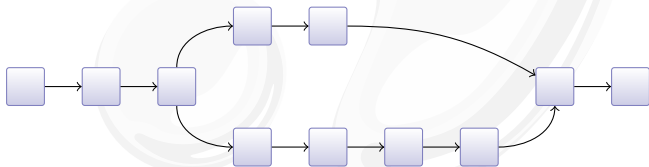
Using History



The Underlying Model

A Mercurial changeset conceptually consist of:

- ▶ 0–2 parent changeset IDs:
 - ▶ root changeset has no parents
 - ▶ normal changesets have one parent
 - ▶ merge changesets have two parents
- ▶ date, username, commit message
- ▶ difference from first parent changeset
- ▶ changeset ID is computed as SHA-1 hash of the above
- ▶ makes it impossible to inject **malicious code** on server



Immutable History

SHA-1 hashes as changeset IDs have some consequences:

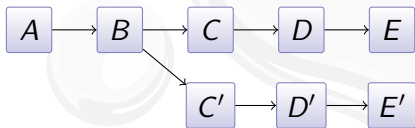
- ▶ a changeset ID is a hash of the entire history
- ▶ changing history changes subsequent changesets
- ▶ history is immutable, you can only make new history:



Immutable History

SHA-1 hashes as changeset IDs have some consequences:

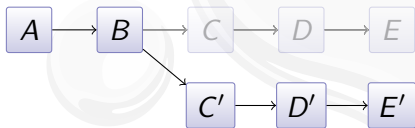
- ▶ a changeset ID is a hash of the entire history
- ▶ changing history changes subsequent changesets
- ▶ history is immutable, you can only make new history:



Immutable History

SHA-1 hashes as changeset IDs have some consequences:

- ▶ a changeset ID is a hash of the entire history
- ▶ changing history changes subsequent changesets
- ▶ history is immutable, you can only make new history:



Outline

Performance Study: OpenOffice

Subversion and Branches

The Underlying Model

Using History



Browsing the History of a File

The `hg annotate` command is invaluable:

- ▶ you see when each line was introduced
- ▶ you can quickly jump back to earlier versions

History of Mercurial's README file:

```
3942: Basic install:
  445:
3942: $ make                # see install targets
3942: $ make install       # do a system-wide install
3942: $ hg debuginstall    # sanity-check setup
3942: $ hg                  # see help
  0:
# ...
```

Better interface in `hg serve`



Searching File Content

Ever wondered when a function was introduced?

- ▶ `hg grep` can help you!

Example: When was `hg forget` introduced?

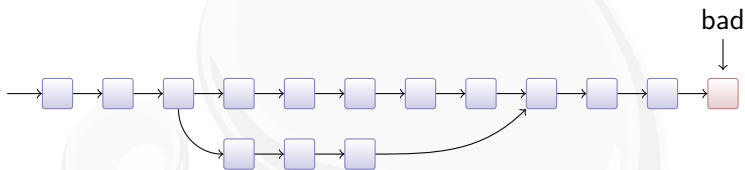
```
$ hg grep --all 'def forget' commands.py
commands.py:8902:+:def forget(ui, repo, *pats, **opts):
commands.py:3522:-:def forget(ui, repo, *pats, **opts):
commands.py:814:-:def forget(ui, repo, file1, *files):
commands.py:814:+:def forget(ui, repo, *pats, **opts):
# ...
```



Revision Graph Bisection

You've found a bug! When was it first introduced?

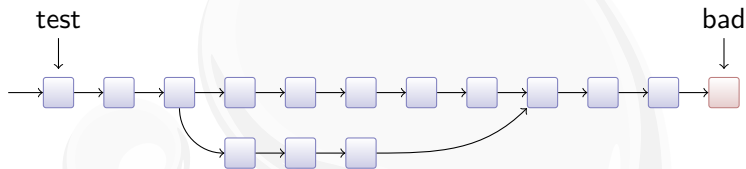
Use `hg bisect` to mark good and bad revisions:



Revision Graph Bisection

You've found a bug! When was it first introduced?

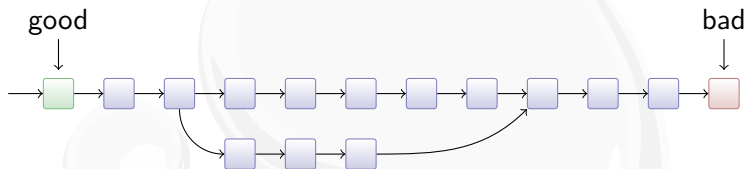
Use `hg bisect` to mark good and bad revisions:



Revision Graph Bisection

You've found a bug! When was it first introduced?

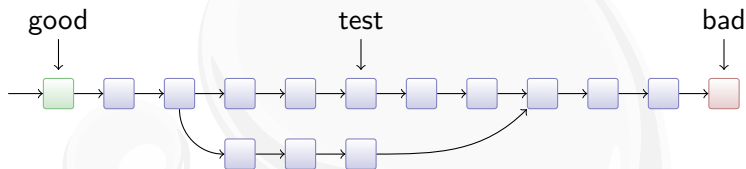
Use `hg bisect` to mark good and bad revisions:



Revision Graph Bisection

You've found a bug! When was it first introduced?

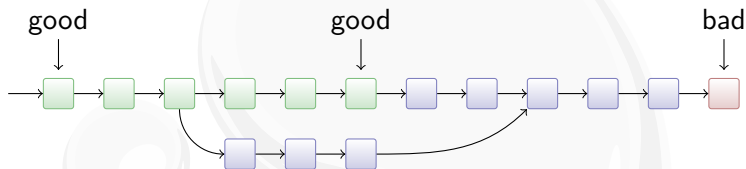
Use `hg bisect` to mark good and bad revisions:



Revision Graph Bisection

You've found a bug! When was it first introduced?

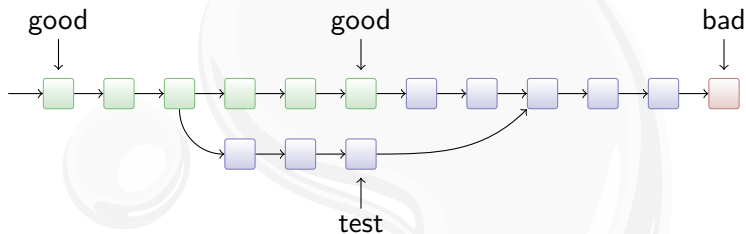
Use `hg bisect` to mark good and bad revisions:



Revision Graph Bisection

You've found a bug! When was it first introduced?

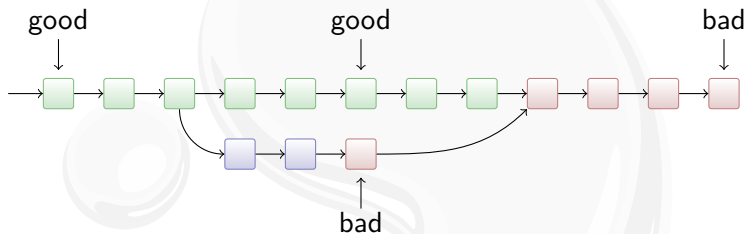
Use `hg bisect` to mark good and bad revisions:



Revision Graph Bisection

You've found a bug! When was it first introduced?

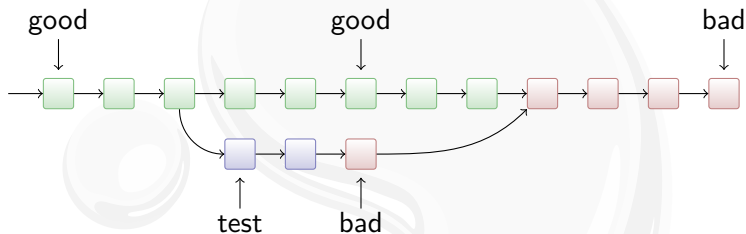
Use `hg bisect` to mark good and bad revisions:



Revision Graph Bisection

You've found a bug! When was it first introduced?

Use `hg bisect` to mark good and bad revisions:



Revision Graph Bisection

You've found a bug! When was it first introduced?

Use `hg bisect` to mark good and bad revisions:

