Distributed Configuration Management: Mercurial

CSCI 5828 Spring 2012

Mark Grebe

Configuration Management

- Configuration Management (CM) systems are used to store code and other artifacts in Software Engineering projects.
- Since the early 70's, there has been a progression of CM systems used for Software CM, starting with SCCS, and continuing through RCS, CVS, and Subversion.
- All of these systems used a single, centralized repository structure.

Distributed Configuration Management

- As opposed to traditional CM systems, Distributed Configuration Management Systems are ones where there does not have to be a central repository.
- Each developer has a copy of the entire repository and history.
- A central repository may be optionally used, but it is equal to all of the other developer repositories.

Advantages of Distributed Configuration Management

- Distributed tools are faster than centralized ones since metadata is stored locally.
- Can use tool to manage changes locally while not connected to the network where server resides.
- Scales more easily, since all of the load is not on a central server.
- Allows private work that is controlled, but not released to the larger community.
- Distributed systems are normally designed to make merges easy, since they are done more often.

Mercurial Introduction

- Mercurial is a cross-platform, distributed configuration management application.
- In runs on most modern OS platforms, including Windows, Linux, Solaris, FreeBSD, and Mac OSX.
- Mercurial is written 95% in Python, with the remainder written in C for speed.
- Mercurial is available as a command line tool on all of the platforms, and with GUI support programs on many of the platforms.
- Mercurial is customizable with extensions, hooks, and output templates.

History of Mercurial

- Development of Mercurial began in 1985.
- It grew out of the need to replace Bitkeeper for maintaining the Linux kernel.
- Matt Mackal started development of Mercurial as a Bitkeeper replacement a few days after Linus Torvalds started the development of git for the same purpose.
- Git was chosen by the Linux kernel team, but Mercurial is used on many open source and commercial development efforts.
- Mercurial is used by many large projects, including Mozilla, OpenJDK, OpenSolaris, Xen and Python.

Getting Started with Mercurial (I)

- Mercurial may be downloaded for most platforms at the web site http://mercurial.selenic.com/downloads/ .
- A Windows version with built in Windows Explorer integration is TortiseHg.
- There are several Mac OSX GUI clients, among them MacHg and SourceTree.
- There are also cross platforms GUI clients such as EasyMecurial.
- In addition, many IDE's support the use of Mercurial, including Eclipse, Visual Studio, and NetBeans.

Getting Started with Mercurial (II)

- The one required step you need to do after install is set your username for Mercurial. This is the default username used for committing changes to the repository. The username is set in the .hgrc file in your home directory:

-
  ```
  [ui]
  username=Mercurial User<Mercurial.User@colorado.edu>
  ```
- The command for all Mercurial activities is hg. (The atomic symbol for mercury – cute huh? ☺)
- To get help on any Mercurial topic just issue the command:
  - `> hg help`

Mercurial Repositories (I)

- Mercurial Repositories are simple directory trees in the file system.
- Mercurial keeps it’s metadata in a .hg subdirectory at the root directory of the repository.
- Since the repository is simply a directory, it may be renamed using normal file renaming techniques.

Mercurial Repositories (II)

- Creating a Mercurial Repository:
  - By giving a directory name (which will be created if it does not exist)
  - `> hg init ~/test_repository`
  - If no name, is given, the current directory will used.
- Cloning a Mercurial Repository
  - Since it is a directory, it could simply be copied using file tools to copy. However, to make sure that the history of what repository it was cloned from, always clone it with Mercurial:
  - `> hg clone ~/test_repository ~/clone repository`

Mercurial Repositories (III)

- Cloning a remote Mercurial Repository
  - A repository can be accessed over http:
  - `> hg clone http://www.mycompany.com/test_repository ~/local_repository`
  - A repository can also be accessed over ssh:
  - `> hg clone ssh:me@you.com/test_repository ~/local_repository`
  - More on sharing repositories later…

Mercurial Working Directories (I)

- A Mercurial working directory is the rest of the repository outside of the .hg subdirectory.
- It is a editable snapshot of some revision of the files stored in the repository.
- Files can be edited using any system tools, and do not require Mercurial commands to edit.
- Any changes made to the files are saved to the repository when you commit (more on that in a bit…)

Mercurial Working Directories (II)

- Files and may be added to the repository with the add command:
  - `> hg add newfile`
- Note, if you add a directory, every file underneath it will be added.
- Files that are not to be controlled (such as derived binaries) can be ignored using:
  - `> hg ignore binaryfile`
- You can check the status of your working directory at any time with the command:
  - `> hg status`

Mercurial Working Directories (III)

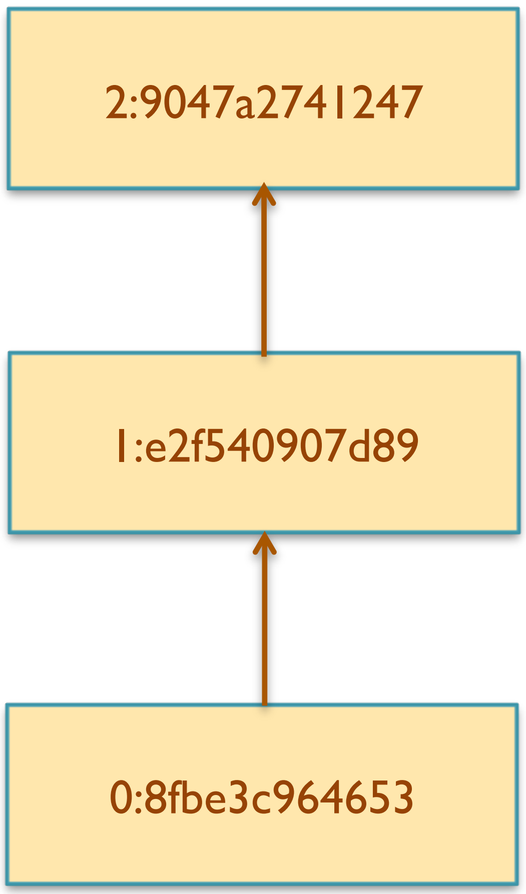
- The output of the status command shows files that have been modified(M), added(A), removed(R), and not tracked in repository(?).

  -
    ```
    > hg status
    A newfile
    ? outside_file
    ```
- Files the are not changed(C) or ignored(I) are not normally shown in the status, but may be by adding –A to the status command.

Mercurial Changesets (I)

- Mercurial identifies the revision "snapshot" of the working directory as a changeset.
- Each changeset is created by making changes to files and then issuing the commit command:
  - `> hg commit -m "Reason for change"`
- The reason for the change may be added to the command line as above, or if left off, the default editor will be started to add the comment.
- If no filenames are specified, ALL changed files will be committed as part of the changeset. To only commit certain files, they must be specified. Note, this may be a surprise for Subversion users.

Mercurial Changesets (II)



- Mercurial tracks changesets with two identifiers, printed as a decimal number:hex number.
- The decimal number is a local identifier. The same version in two different repository clones may have different local ids.
- A 40 digit hex number which is globally unique (only the first 12 digits are normally printed). The same version in different repository clones with have the same hex ID.
- Each changeset has at least one parent changeset (except for the first). A changeset that does not have any childrean is called a "head".

Mercurial Changesets (III)

- Information about the changesets in a repository may be displayed with the log command:

```
◦ > hg log
◦ changeset:   1:e2f540907d89
◦ tag:         tip
◦ user:        Mercurial User <Mercurial.User@colorado.edu>
◦ date:        Mon Mar 12 20:48:50 2012 -0500
◦ summary:     The second revision

◦ changeset:   0:8fbe3c964653
◦ user:        Mercurial User <Mercurial.User@colorado.edu>
◦ date:        Mon Mar 12 17:51:32 2012 -0500
◦ summary:     Initial version!
```

- Note that the user who made the change is tracked.
- If –v is added to the command, the files which changed in each changeset will also be displayed. Also if you want the history of only a specific file, it should be listed on the command line.

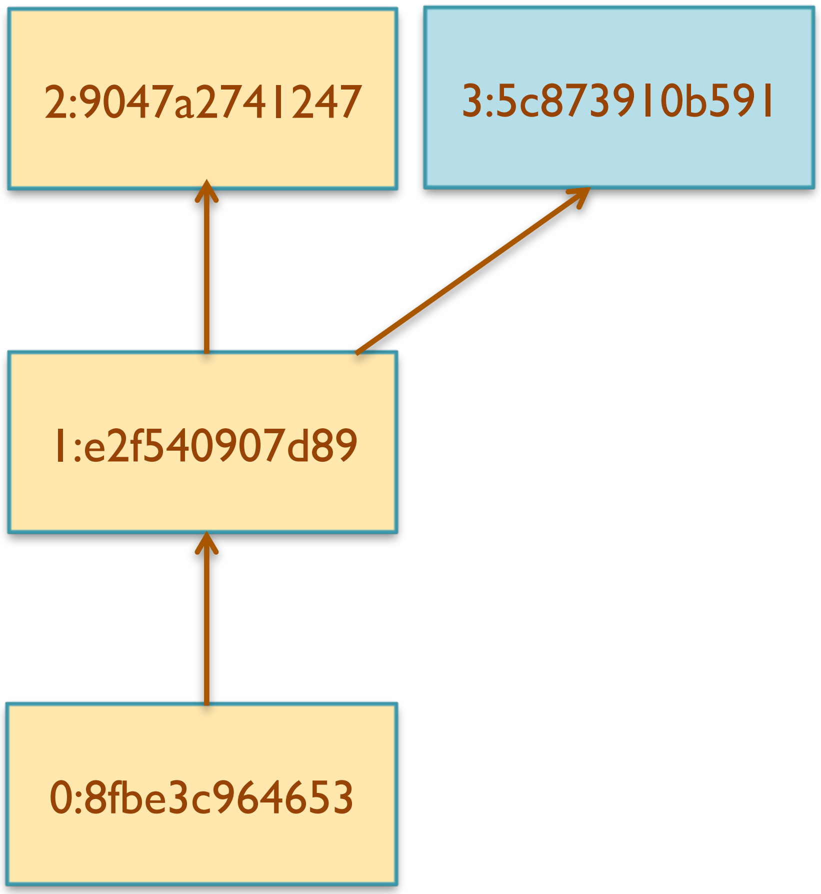
Remembering a Changeset

- Changesets may be labeled with a tag, which is useful for marking releases, etc.
- Tags in Mercurial work somewhat differently than other CM systems.
- Tags are stored in a file in the working directory called .hgtags. Because of this, tagging a changeset will actually create a new changeset to commit the changes to .hgtags.
- There is a special tag called 'tip' which always refers to the newest changeset in the repository.

Examining a Historical Changeset

- You can compare a file from two changesets with the diff command, which will output in standard Unix diff format (or with a Mercurial extension, you can use your favorite external diff tool). The following compares test.txt from revisions 2 and 5:
  - `> hg diff -r 2 -r 5 test.txt`
- Also, at any time you can move your working directories parent to an older changeset and explore the files directly with native tools:
  - `> hg update 3`
- The above command makes the working directories parnent to local changeset id 3, but you can also use a tag or a hex id as a parameter.

Mercurial Branches



- What happens if you make changes while your working directory has a parent other than the tip, and then do a commit?
- As usual, a new changeset will be created, however it's parent will be the changeset your working directory had as a parent.
- In other words, it will create a branch, and there will now be two heads.
- There are two methods of labeling branches, temporary bookmarks and permanent branch names

Sharing Changes - Pull

- You may get changes from another repository by doing a "pull" of the changes. The other repository may be local, remote via http(s), or remote via ssh.
  - `> hg pull ~/other_repository`
- Note, that by default the pull operation does not update the working directory. To do this you must issue an update command, or add –u to the pull command.
- If you would like to see what changes you would get from the other repository, but not actually retrieve them, then instead do:
  - `> hg incoming ~/other_repository`

Sharing Changes - Push

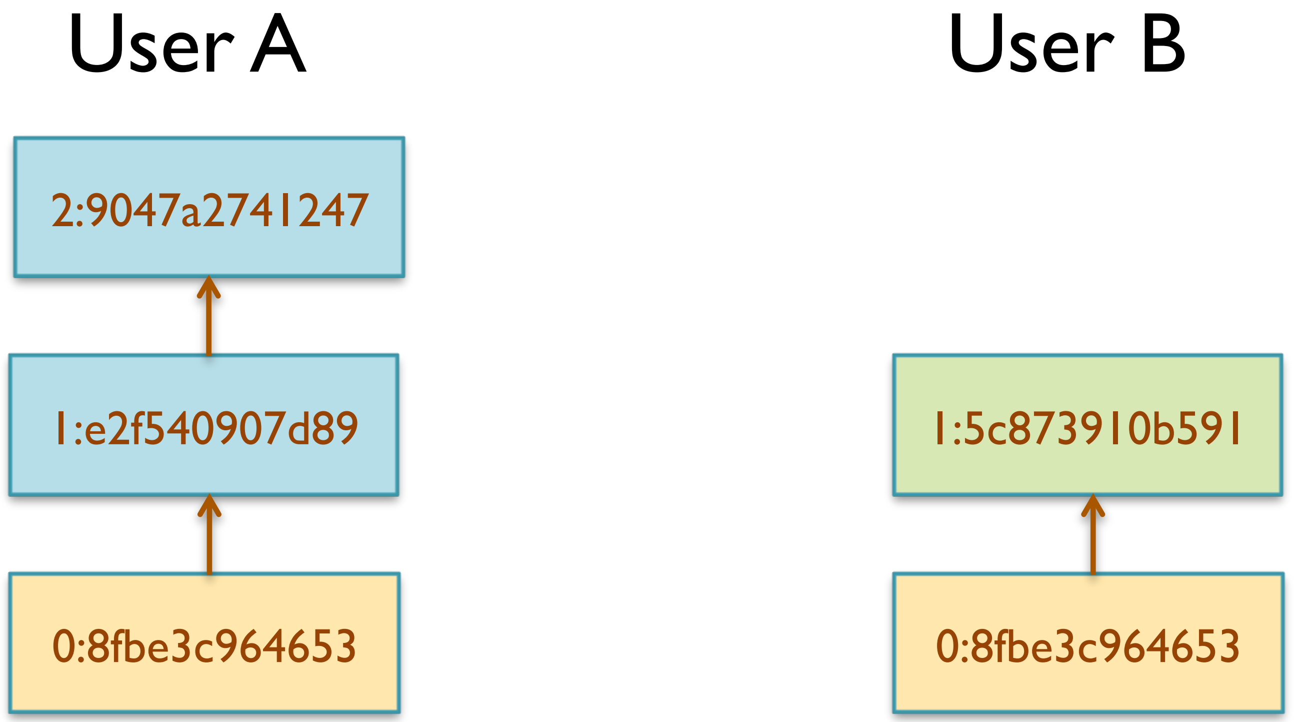
- You may send changes to another repository by doing a "push" of the changes. The other repository may be local, remote via http(s), or remote via ssh.
  - `> hg push ~/other_repository`
- Note, again that the push operation does not update the working directory in the repository you are pushing to. An update command must be issued there. There is no –u option to the push, which makes sense, since you are modifying some other repository.
- If you would like to see what changes you would send to the other repository, but not actually send them, then instead do:
  - `> hg outgoing ~/other_repository`

Sharing Changes – Web Server

- Mercurial provides convenient repository web publish. To start a temporary web server
  - `>hg serve`
- This basic server does not provide authentication. Options allow you to specify if pushes are allowed (they aren't by default). For a more secure and permanent solution, the Mercurial distribution provides a CGI script to use with other web servers.

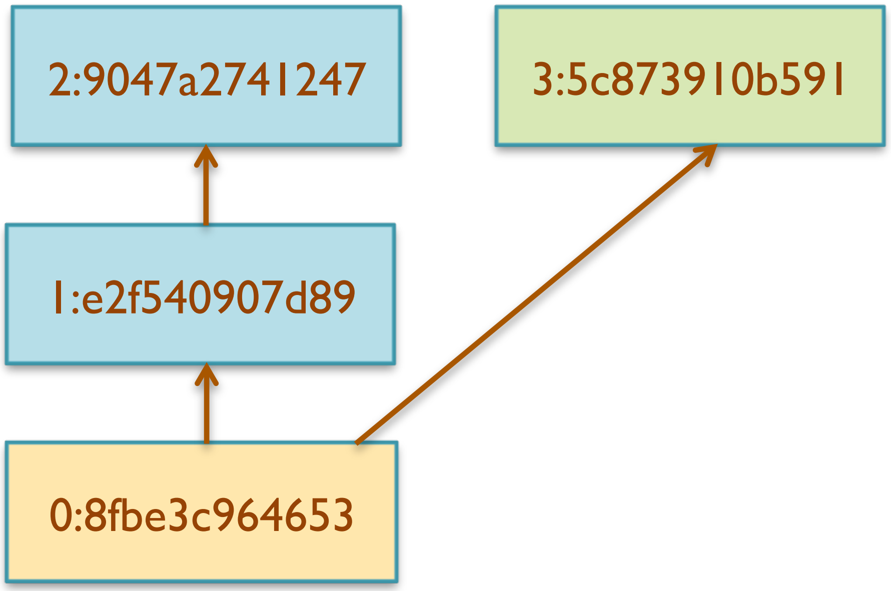
Mercurial Merges(I)

- Two users start from a common repository that they both clone, and then proceed to make changes:



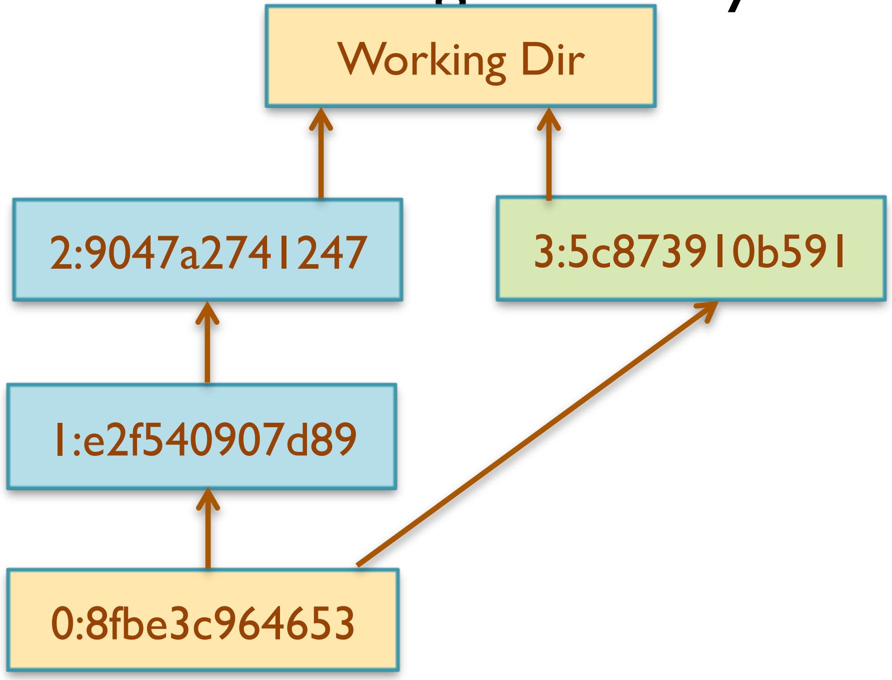
Mercurial Merges (II)

- User A "pulls" the changes from user B's repository:
  - `> hg pull /home/userB/myRepo`



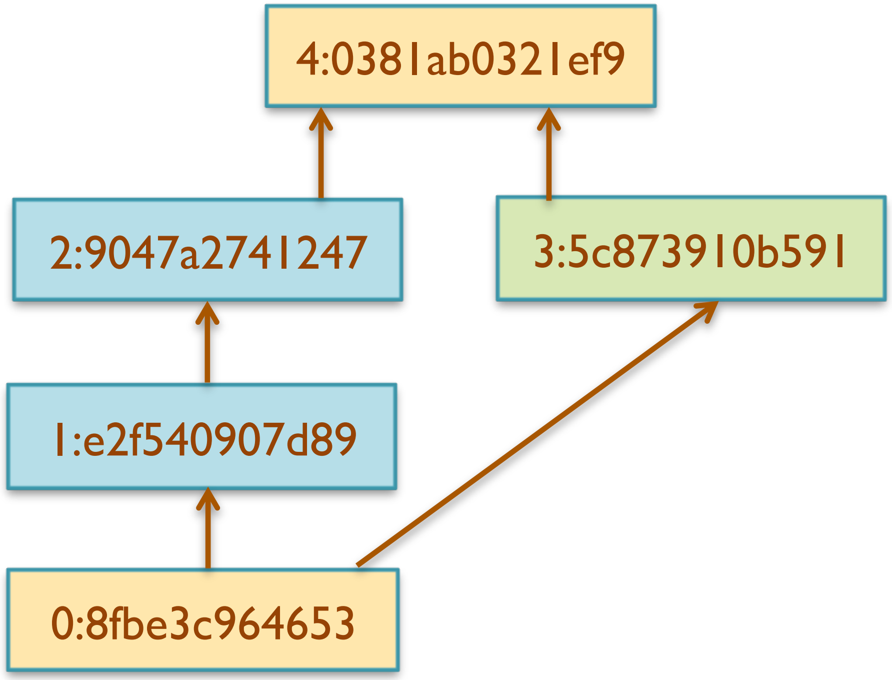
Mercurial Merges (III)

- User A merges the changes:
  - `> hg merge`
- Mercurial will merge the changes automatically, or if it is not able to, will invoke a external merge tool to allow user intervention, and save the changes in the working directory.



Mercurial Merges (IV)

- User A commits the changes:
  - `> hg commit`
- This creates a new changeset which has both branches as parents. The merged changes could be pushed to User B, or User B could pull the changes from User A.



Moving Data Between Mercurial and Other CM's

- Mercurial comes with an extension, convert, that allows you to import history from other CM systems, including Subversion, CVS, Git and Bazaar.
- In addition, with Subversion, you can move history in both directions to enable Subversions users to try Mercurial before they "commit" to it.

Mercurial vs Git

- As of this writing, Git is likely the most used Distributed CM system, with Mercurial being second.
- There are many resources on the web providing comparisons of these two systems (many of them, however, verge on being religious diatribes).
- Mercurial will feel more comfortable to longtime CVS and Subversion users.
- At the end of the day it comes down to what someone likes. I would recommend you try both.

References

- This presentation has given an overview of Mercurial usage. The following are excellent options for further investigation:
  - O'Sullivan, Bryan. Mercurial: The Definitive Guide, 2009. (Available online at http://hgbook.red-bean.com/)
  - http://mercurial.selenic.com/wiki/Tutorial
  - http://mercurial.selenic.com/wiki/UnderstandingMercurial

Example Usage

- Accompanying this slide presentation is a screencast which demonstrates using Mercurial.