Bringing Undo to system admin: a new paradigm for recovery

Aaron Brown
UC Berkeley CS Division
abrown@cs.berkeley.edu
http://roc.cs.berkeley.edu
Motivation

• Recovery is important
  - people screw up
  - software and hardware break
  - upgrades fail
  - hackers break in
  - etc.
  - and sysadmins have to clean up the mess
    » can we make life easier?
What makes recovery easy?

- **Not having to think about it beforehand**
  - do you have a backup strategy to handle your typos?

- **Having a consistent strategy system-wide**
  - no trying to disambiguate user/system data

- **Being familiar with it**
  - recovery: it’s not just for catastrophes anymore
  - easy recovery => more freedom to experiment, learn

- **Not having to do it at all**
  - export recovery to users

- **This is not what we have today!**
Undo: a new recovery paradigm

- Make system recovery as painless and natural as undoing mistakes in a word processor

- Continuous recovery with undo: the 3 R’s
  - **Rewind**: roll system state backwards to any time point
  - **Repair**: fix problem; reconfigure to avoid problem
  - **Redo**: roll system state forward, replaying user interactions lost during rewind
Undo makes recovery easy

• No explicit definition of recovery points

• Covers system and user data
  - repair corruption, virus damage, trojans, ...

• Redo means no loss of user data on rollback

• Provides forgiving environment
  - encourages learning via experimentation

• Can export to users
Status

• **Now:** defining the conceptual model
  - input welcome! would undo improve your life? where would you like to see it?

• **Next:** studying implementation techniques
  - no-overwrite storage
  - logging of state and user actions
  - using dependencies between state to guide rollback

• **Goals:**
  - proof-of-concept implementation (email service)
  - set of design guidelines for building undo-recoverable systems
  - if possible, an API and infrastructure for undoable systems
Contact

Aaron Brown, UC Berkeley
abrown@cs.berkeley.edu

This work is part of the ROC (Recovery-Oriented Computing) Project, run by Dave Patterson

http://roc.cs.berkeley.edu