Towards A Better SCM: Revlog and Mercurial

Matt Mackall
Selenic Consulting
mpm@selenic.com
Things To Consider When Building A New SCM
Things To Consider When Building A New SCM

- Scalable
Things To Consider When Building A New SCM

- Scalable
- Atomic
Things To Consider When Building A New SCM

- Scalable
- Atomic
- Decentralized
Things To Consider When Building A New SCM

- Scalable
- Atomic
- Decentralized
- Convenient Branching
Things To Consider When Building A New SCM

- Scalable
- Atomic
- Decentralized
- Convenient Branching
- Repeated Merge
Things To Consider When Building A New SCM

- Scalable
- Atomic
- Decentralized
- Convenient Branching
- Repeated Merge
- Robust Storage
Things To Consider When Building A New SCM

- Scalable
- Atomic
- Decentralized
- Convenient Branching
- Repeated Merge
- Robust Storage
- Easy to Use
Things To Consider When Building A New SCM

- Scalable
- Atomic
- Decentralized
- Convenient Branching
- Repeated Merge
- Robust Storage
- Easy to Use
- Portable
Early History Of Mercurial
Early History Of Mercurial

• April 6, 2005: Bitmover announces end of gratis version of Bitkeeper
Early History Of Mercurial

• April 6, 2005: Bitmover announces end of gratis version of Bitkeeper
  Linus mentions he's looking at alternatives
Early History Of Mercurial

• April 6, 2005: Bitmover announces end of gratis version of Bitkeeper
  Linus mentions he's looking at alternatives
  I start working on Mercurial
Early History Of Mercurial

- April 6, 2005: Bitmover announces end of gratis version of Bitkeeper
  Linus mentions he's looking at alternatives
  I start working on Mercurial
  Linus starts working on Git
Early History Of Mercurial

- April 6, 2005: Bitmover announces end of gratis version of Bitkeeper
  Linus mentions he's looking at alternatives
  I start working on Mercurial
  Linus starts working on Git

- April 8: Linus releases initial nearly useless Git snapshot
Early History Of Mercurial

- April 6, 2005: Bitmover announces end of gratis version of Bitkeeper. Linus mentions he's looking at alternatives.
- I start working on Mercurial.
- Linus starts working on Git.

- April 8: Linus releases initial nearly useless Git snapshot.

- April 19: Mercurial 0.1 released. Features: familiar interface, efficient storage, commit/checkout/clone/pull/merge.
Early History Of Mercurial

- April 6, 2005: Bitmover announces end of gratis version of Bitkeeper
  Linus mentions he's looking at alternatives
  I start working on Mercurial
  Linus starts working on Git
- April 8: Linus releases initial nearly useless Git snapshot
- April 19: Mercurial 0.1 released features: familiar interface, efficient storage, commit/checkout/clone/pull/merge
- April 20: Linus fails to destroy Git in a timely fashion
Desirable Properties For Revision Storage
Desirable Properties For Revision Storage

- $O(1)$ addition and retrieval
Desirable Properties For Revision Storage

- O(1) addition and retrieval
- immutable or append-only
Desirable Properties For Revision Storage

- O(1) addition and retrieval
- immutable or append-only
- decent compression
Desirable Properties For Revision Storage

- O(1) addition and retrieval
- immutable or append-only
- decent compression
- strong integrity checks
Desirable Properties For Revision Storage

- $O(1)$ addition and retrieval
- immutable or append-only
- decent compression
- strong integrity checks
- cluster file changes together on disk
Desirable Properties For Revision Storage

• O(1) addition and retrieval
• immutable or append-only
• decent compression
• strong integrity checks
• cluster file changes together on disk
• efficient logging and annotate
Revlogs

- revlog index
  - revision 6 record
- revlog data
  - full revision 4
  - delta 4-5
  - delta 5-6
Changesets, Manifests, and Files

Diagram:
- Changeset Index
  - Changeset Data
  - Linkrev
  - Manifest Index
  - Manifest Data
  - File Index
  - File Data
Transactions and Rollback
Transactions and Rollback

• Every repository write is protected by a simple transaction log
Transactions and Rollback

- Every repository write is protected by a simple transaction log
- The log records the starting length of each revlog touched
Transactions and Rollback

- Every repository write is protected by a simple transaction log
- The log records the starting length of each revlog touched
- On abort, each revlog is truncated to its original length
Transactions and Rollback

- Every repository write is protected by a simple transaction log
- The log records the starting length of each revlog touched
- On abort, each revlog is truncated to its original length
- We save the most recent transaction log to allow manual rollback (“undo”)

Synchronization and Merging

1. Alice makes changes
2. Bob makes changes
3. Alice pulls changes from Bob's repository
4. Alice merges with Bob
Taking Advantage of the FS
Taking Advantage of the FS

- Avoiding seeks is critical for performance
Taking Advantage of the FS

- Avoiding seeks is critical for performance
- Traversal order matters!
Taking Advantage of the FS

- Avoiding seeks is critical for performance
- Traversal order matters!
- Ordering by hash means random seeking in the working directory and degrades to random seeking on copy
Taking Advantage of the FS

- Avoiding seeks is critical for performance
- Traversal order matters!
- Ordering by hash means random seeking in the working directory and degrades to random seeking on copy
- Ordering by modification time degrades to random seeking over time
Taking Advantage of the FS

- Avoiding seeks is critical for performance
- Traversal order matters!
- Ordering by hash means random seeking in the working directory and degrades to random seeking on copy
- Ordering by modification time degrades to random seeking over time
- Ordering by pathname is stable and gives largely monotonic head movement
Some Other Optimizations
Some Other Optimizations

- Mercurial uses a custom delta algorithm
Some Other Optimizations

- Mercurial uses a custom delta algorithm
- Applying long chains of deltas is clever
Some Other Optimizations

- Mercurial uses a custom delta algorithm
- Applying long chains of deltas is clever
- Careful ordering avoids locking for most operations
Some Other Optimizations

- Mercurial uses a custom delta algorithm
- Applying long chains of deltas is clever
- Careful ordering avoids locking for most operations
- Local clones use copy-on-write
Some Other Optimizations

- Mercurial uses a custom delta algorithm
- Applying long chains of deltas is clever
- Careful ordering avoids locking for most operations
- Local clones use copy-on-write
- Remote clone uses recompression for WAN transmission
Some Other Optimizations

- Mercurial uses a custom delta algorithm
- Applying long chains of deltas is clever
- Careful ordering avoids locking for most operations
- Local clones use copy-on-write
- Remote clone uses recompression for WAN transmission
- Network protocol uses graph discovery algorithm for efficiency
A Benchmark
A Benchmark

- commit 773 patches (20MB) for 2.6.18-rc1 to -mm2
A Benchmark

- commit 773 patches (20MB) for 2.6.18-rc1 to -mm2
- 1.8GHz AMD64 laptop, 1.2GB of RAM freshly formatted ext3 filesystem, data=writeback,noatime
A Benchmark

- commit 773 patches (20MB) for 2.6.18-rc1 to -mm2
- 1.8GHz AMD64 laptop, 1.2GB of RAM freshly formatted ext3 filesystem, data=writeback,noatime
- Git 1.4.1:
  
  ```bash
  $ git-quilt-import 2.6.18-rc1-mm2
  real: 2m7.701s user: 1m15.953s sys: 0m30.186s
  ```
A Benchmark

- commit 773 patches (20MB) for 2.6.18-rc1 to -mm2
- 1.8GHz AMD64 laptop, 1.2GB of RAM freshly formatted ext3 filesystem, data=writeback,noatime
- Git 1.4.1:
  
  $ git-quilt-import 2.6.18-rc1-mm2
  real: 2m7.701s user: 1m15.953s sys: 0m30.186s

- Mercurial:
  
  $ hg qpush -a 2.6.18-rc1-mm2
  real: 1m18.398s user: 0m42.511s sys: 0m10.105s
Mercurial Wiki:
http://selenic.com/mercurial