Mercurial 3.4 Sprint Notes Documentation

Release 0.1

Sean Farley, Pierre-Yves David, and Augie Fackler

Nov 18, 2017

Contents

Contents:

CHAPTER 1

General

- How to handle new features for OSS projects (basically detecting client versions)? eg: mozilla may enforce using bundle2 at some point.
- Manifest Storage: change other things at same time as manifestv2/tree-hashes?
  - How to handle differing manifest/changelog versions?
  - Since manifestv2 changes, hashes, time for a changelogv2?
- `.hgtags` and `.hgignore` with narrow checkouts?
- Commit signing
- Obsolesence marker discovery
- Evolve UI discussion
- Handling evolve divergence
- Including "reflog" in core
- Remote bookmarks in core
- Automated package building for python 2.4
  - rpms with embedded python
  - OS X builds
- Stricter author field validation
  - Conclusion: best left as a commit hook on the server

CHAPTER 2

Patch Review

We need to:

- Have a single source of Truth (unify: inflight, patchbot, patchwork)
- A way for non-email-nerd to review patches
- A way for new people to submit patches
- A way to track multiple version/comment on patch
- Herald rules! (automatic triage)

We need a two-way synchronization:

- Things will get bad so they can get better,
- We need to put more actual resources on that.

CHAPTER 3

Manifest discussions

3.1 Day 1

- 'Schema' is what is exchanged on the wire protocol. It's not been broken in the past, so that old clients can still work with it.
- Tree manifests: It's possible to calculate both hashes (tree hash and flat manifest hashes).
- Mozilla central needs 25MB of mapping from old to new hash scheme for both changeset and manifest
- Having the mapping might cause people to reimplement `git alias` (spectral note: I don't know what this is ;))
- Two switches: new manifest format, and new manifest hash
  - Mozilla can use the new format (trees, manifest v2) without a new hash
- While we're changing the 'schema' (hashes), what do we want from changeset v2?
  - "extra" key/value pairs on individual lines
  - Support n in filenames
  - Rename information?
  - add/edit/delete status in changesest (no need to touch manifest)?
  - More strict author field validation (require email/rfc format?)
- `hg log` on directories becomes faster with tree manifests
- `rename cache` to store that a file has not been renamed, so that a lot of checks become much quicker
- Historical note: the reason for the file list in the changeset is that it's for push/pull, so deletes didn't originally show up there, because it didn't change the filelog.
- sid0: do we want to store 'this got deleted' information in the filelogs, so that `hg log <file>` shows that it happened?
- Default is 'flat manifests' since gut-feel is ~98% of projects this is the best one for them

- Certain projects want tree manifests on disk (client? or server? or both (separately? concurrently?))
- Could make a read-only copy using old hash to do a more gradual migration to exchanging tree manifests?

Three use cases:

1. Mozilla central today: flag to turn on that uses a new disk format, but no hash changes, so exchange is unaffected.
2. Google soon: start from scratch with new hashes
3. Transition from 1->2

Two flags:

1. Storing tree manifests locally (old hashing)
2. Break the schema

For flag #1 without #2: The manifest revlog (root-level 00manifest.{i,d}) would have the old hash as its nodeid, and it wouldn't strictly match the contents at that version.

An extension (client and server side) that can maintain a map for old-hashes in bug trackers?

For getting to Flag #2:

- Default on the server is that it does not accept manifest v2
- no v1 children with v2 parents
- Server then enables v2 pushes to it, the next change with v2 will upgrade all future changes
- Upgrade during exchange v1->v2? Maybe not needed?
- Command to downgrade from v1->v2 if you get 'infected' with the virus should be pretty easy.
- flat-hashing a tree manifest would be more difficult than it might seem at first, because parent revisions
- A new challenger appears! (4th use case?)
- Matrix: flat-right-now vs. flat-with-subdir-hashes vs. tree manifests, manifestv1 vs. manifestv2, hashv1 vs. hashv2
  - Are deltas going to be broken in any of these?
  - Manifest Feature Maxtrix
  - So we're thinking implement 6, 8/9, 14 on the way to 17, benchmark them, see if the benefits make it so that implementing the conversion-during-exchange makes sense.
    - benchmarks need to consider clone time, server cpu usage, on-disk size
    - 6=14 and 8=17 if we don't care about breaking hashes, 8=9 if we don't care about exchange
- Client version announcement (User-Agent string?)
  - As a 'backport extension'?
  - Include hg version, extensions? python version? platform?

3.2 Day 2

Google wants new tree-structure manifests.

It'd be nice to not break old clients. Can compute old format hash for tree manifest on disk.

Three use cases:

1. mozilla-central today
2. Google soon
    - Never accept v1 manifests, ever.
3. Transition from 1 to 2 case
    - (~2 years out, needs time for clients to upgrade naturally)
4. prevention use case
    - Implementation-wise, this really means you don't set the schema change flag on the server.
    - Idea: server could rewrite as v1 when receiving push using v2, tell client (using bundle2)

Two flags:

1. Store tree manifests locally but use old hashing
    - Transcode to old manifest format over the wire
    - store old hash in the changelog entry
2. Break the schema
    - allow new hashing scheme to be recorded in changelog
    - exchange the new revlogs

MAY enforce a changeset schema change when we do flag 2? Not sure if it really matters.

Layout v2: orthogonal from all of these concerns?

- Puts file hashes on separate lines for compression benefits

CHAPTER 4

Manifest Feature Matrix

Original spreadsheet

Num	Hash	Client On-Disk Manifest Format	Client Tree Manifest On-Desk	Tree Manifest in Exchange	Consequences: Read-delta works	Sane?	Use Case	Benefits	Interesting?
2	Current	V1	No	No	Yes	Yes	Existing Projects	Mostly read-delta	Obviously
6	Current	V2	No	No	No	?		Size: 30% smaller without general delta	?
8	Current	V2	Yes	No	No	Yes	Mozilla	Rebase et. al faster on client, old clients won't break	?
9	Current	V2	Yes	Yes	No	Kind of	Exchange for modern client and server in above case		
14	Tree	V2	No	No	No	Maybe	Small new project	Compact representation, less disk seeks	
17	Tree	V2	Yes	Yes	No	Yes	Google	Narrow clones	Obviously

- Next step: analyze storange and perf of 14 and 17 on normal-size and mozilla-size repos to see if we should support 6 and 8.
- Concern: if exchange uses v1 format and disk uses v2, we have to do transformation between formats to apply deltas.

- If we can’t do old client compat, then we should only do row 2 and 17
- New delta encoding might also be worth considering, but completely orthogonal to this.

CHAPTER 5

Bikeshed Discussion

Long-standing issues about what functionality to bring into core.

- (Approved) Facebook's reflog extension
  - renamed to 'journal'
- (Approved) progress bar (held up by bug; assigned to Augie)
- (Approved) color in core (256 color patches will be accepted)
- (Approved) pager in core (held up by editor / piping bugs)
- (Approved) backups (finding and restoring bundles)
  - rename to something but what?
  - probably as a flag to unbundle
- (Approved) smart log
  - eliding / ellipses in graph
  - topological sorter
  - revset
  - new template
- (Approved) templates
  - new, easy to discover templates needed
  - oneline, twoline, etc.
- (Possibly) share extension
  - everything on by default
  - hg clone –share?
- (Approved) new paths
  - needs to respect `[auth]` sections

  - path aliases
  - Use `[uri]` section for naming?
- (Approved) remote bookmarks
  - built on top of `journarl` and `new paths`
  - can be used to propagate deletion (using a `merge`-like operation)
  - change `hg update` to `hg update -B`?
- (Possibly) terse status
  - needs discussion on the mailing list after 3.4 code freeze
- (Approved) `hg config -l 'section.name = value'`
  - after much, much, much discussion mpm could agree to "doing the dumbest thing possible" -> appending to the end of the .hgrc file

CHAPTER 6

Indices and tables

- genindex
- modindex
- search