Notedb
What? Why? How?

Gerrit User Summit 2015
Dave Borowitz <dborowitz@google.com>
Notedb: Gerrit 3.0
Notedb: New Gerrit storage backend

What?
Why?
How?
Notedb: New Gerrit storage backend

**What?**

**Why?**

**Why?**

**What?**

**How?**
**Notedb: New Gerrit storage backend**

- **What?**
- **Why?**
- **Why?**
- **What?**
- **How?**
Notedb: Why?
Why?

Simplicity  Consistency  Features
Simplicity

- Gerrit administrators need to be database administrators too
Consistency
Consistency

- git push HEAD:refs/for/master
  - a. write to refs/changes/34/1234/5
  - b. update PatchSets table to point (1234,5) to new change
  - c. update Changes table to set current patch set 5
- (b) and (c) are a transaction
- Failures leave a partially-applied state
Consistency

- Solution: store everything in one database!
  - Git?
  - SQL?
Consistency

- Pushes aren’t the only problem
- Replication
- Backups
Features: Federation

- Gerrit is used within many different organizations
- Example: Android
  - Internal fork
  - AOSP
  - Partners
Features: Federation

- Moving changes between servers is hard
- Can only move git data, not code review metadata
Features: Offline Code Review

- Can’t review on an airplane
- Tools need to know REST API
  - Offline support is even more work
Can we standardize what a code review “looks like”?

{Github, Phabricator, ...} are unlikely to support Gerrit’s REST API.
Notedb: What?
Note + DB

- Store review data in git notes
Note + DB

- Store review data in git notes
commit 218825e1e3b44a403be3a79242dfbc0591435223
Author:     A Reviewer <101@gerrit>
AuthorDate: Wed Jun 4 08:31:30 2014 -0700
Commit:     Gerrit Code Review <noreply-gerritcodereview@google.com>
CommitDate: Wed Jun 4 08:31:30 2014 -0700

Review patch set 1

Mostly looks good, but please fix a few things.

Patch-set: 1
Label: Code-Review=-1
Notedb Format

- One (more) ref per change
  - refs/changes/34/1234/meta
- One git commit per operation
  - Audit logging for free
- Simplest possible Key: Value format
- Read full note history on each request
  - Optimized parsing
  - Good block cache locality
Notedb Format

- Where are the notes?
Notedb Format

- Where are the notes?
- Annotate patch set SHA-1 with inline comments
Notedb Format

Patch-set: 1
File: file1
1:1-2:1
Wed Sep 30 14:00:05 2009 -0700
Author: Other Account <2@gerrit>
UUID: uuid1
Bytes: 9
comment 1

File: file2
3:1-4:1
Wed Sep 30 14:00:07 2009 -0700
Author: Other Account <2@gerrit>
UUID: uuid3
Bytes: 9
comment 3
Notedb Format

- Non-Change entities: handwaving!
Notedb Solves Problems

- Simplicity/consistency: everything in one data store
  - Upstream support for atomic ref transactions “coming soon”
- Federation: ship meta refs around
  - Change number agnostic (accounts are trickier)
- Offline: built-in offline storage
Notedb: How?
How are we doing?

- Some tables migrated
  - ChangeMessages, PatchSetApprovals, PatchLineComments
- Others “in progress”
  - Changes, PatchSets
How are we doing it?

- Incrementally
- All code paths update **both** databases
- Reads/writes can be turned on independently
  - Parallel testing
  - Online migration for googlesource.com
Online Migration?!?

- googlesource.com has always done zero-downtime upgrades
- Multi-master setup with rolling updates
  - Always have multiple server versions running
- Basic process:
  - Write to both locations; read from old DB
  - Batch migrate old data
  - Flip the switch to read from new DB
How are we doing it?

- New abstraction: BatchUpdate
- Handles:
  - multi-change transactions
  - writing to git before DB
  - retry/fast-forward semantics
  - notedb/database primary
- Rewriting lots of codepaths
- Helpful independent of notedb
How can you help?

- Coding: convert the last few tables
- Coding: help fix broken tests
- Volunteer for early performance testing
Questions?