Google

Reftable

Faster Git-as-a-Database

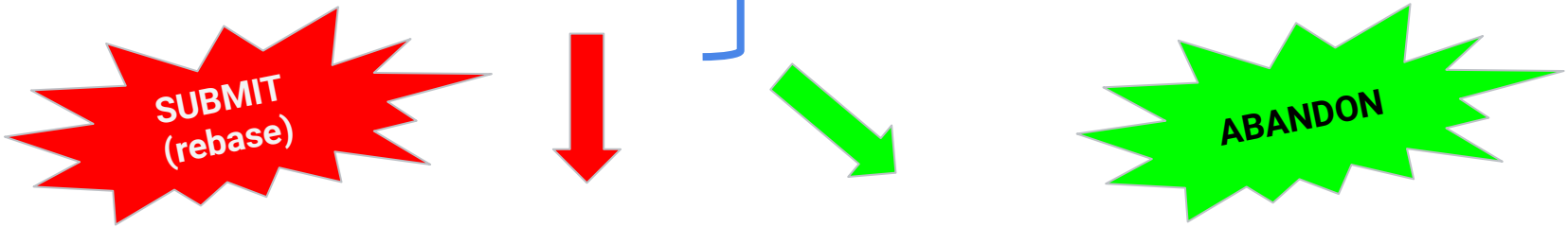
Han-Wen Nienhuys

Agenda

- NoteDB, or Git-as-a-Database
- Ref storage - the Problem
- Reftable: All Hail The King
- Demo
- Outlook

What is a review?

- Code - refs/changes/12345/20
- Metadata
  - Status = active
  - PatchSet = 20



- Create refs/changes/12345/**21**
- Advance refs/heads/master
- Metadata
  - Status = **merged**
  - Patchset = **21**

- Code - refs/changes/12345/20
- Metadata
  - Status = **abandoned**
  - Patchset = 20

A Tale of Two Storage Systems

NoteDb: store metadata in Git branches

- refs/changes/12345/20
- refs/changes/12345/meta

Reasons: with two storage systems:

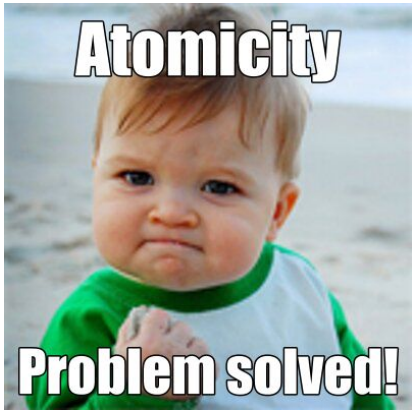
- Consistent operations?
- Concurrent writes?
- Takeout?
- Backup?
- Two systems to maintain, tune, etc.

NoteDb occasion:

“Hey Gerrit team, we’re shutting down your SQL database January 1, 2018.

Rewrite all your glue code, or else.”

NoteDb solves atomicity woes



- Code - refs/changes/12345/20
- Metadata - refs/changes/12345/meta
  - Status = active
  - PatchSet = 20

- Create refs/changes/12345/**21**
- Advance refs/heads/master
- Advance refs/change/12345/meta
  - Status = **merged**
  - Patchset = **21**

Atomic ref update

Git branch access patterns

Read:

- Random access:
  - refs/changes/12345/meta
  - refs/heads/master
- Prefix search:
  - refs/heads/*
  - refs/changes/12345/*

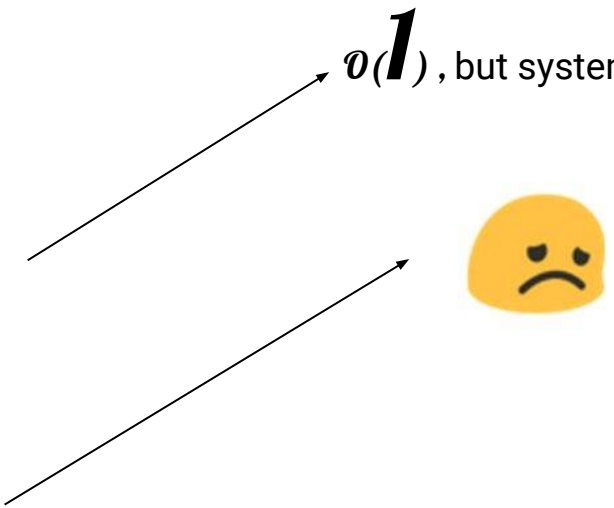
Size:

- Chromium src: 1.7M branches
- Android pfb: 1.8M branches

Branch storage, v1: “loose refs”

File: `.git/refs/heads/master`

- Random access:
  - Open, Read/Write, Close
- Prefix search: refs/changes/*
  - Open, Read, Close directory
  - Recurse
- Space: 4096 bytes / ref , 1 file / ref



File system limitations:

- master/bla vs master
- MASTER vs master

Branch storage, v2: “packed refs”

File: `.git/packed-refs`

```
refs/heads/a    abc123abc..
refs/heads/b    456abc123..
..
refs/heads/z    789def456..
```

- Random access
  - O(log N) in memory
- Prefix search
  - O(log N) in memory
- Space
  - 20 bytes + name / ref
- Write
  - Rewrite file, O(N)

Branch storage, v3: “packed refs” + “loose refs”

- Combine packed & loose
- Loose ref overrides packed-ref
- Compact occasionally

- Performance
  - Packed-refs + a few reads
- Space:
  - 20 bytes + name / ref
- Write
  - write 1 file, O(1)
- Difficult to understand
- Deletion is O(N)

Atomicity in the file system

Lock = create single file

Transaction = rename file to destination

- Recompact loose refs
- Update ref data in memory
- Write new packed-refs file

Cost: O(N)

Google

Fix all of this

- We only need a format for fast reading
- Fast writing
  - transactions are small
  - Write just the delta
  - Merge delta on read
  - Compact regularly
- List of tables is a single file
  - can provide transactions

```
table-1.ref
table-2.ref
table-3.ref
```

```
table-1.ref
..
master:     abc1234
next:       77a2fde
stable:     def4678
...
```

```
table-2.ref

master:     666666
```

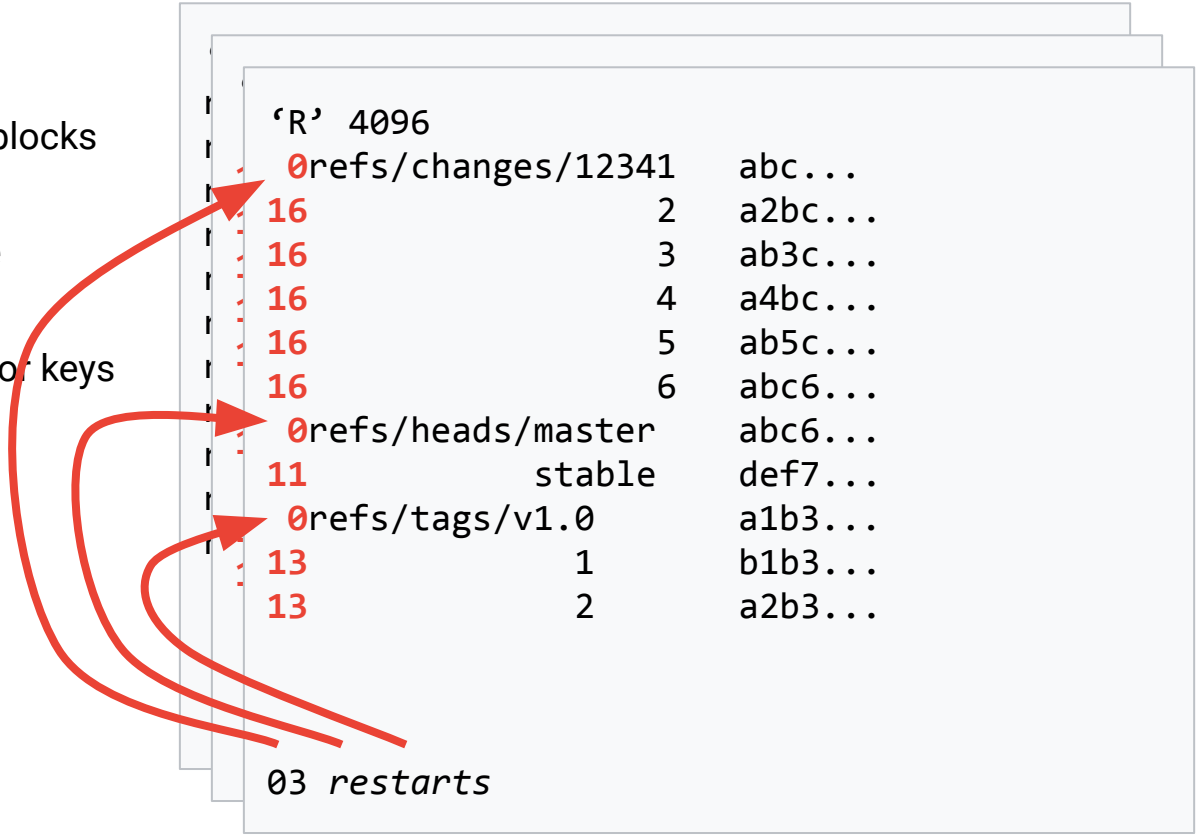
```
table-3.ref

next:       delete
```

```
Merged table
..
master:     666666
stable:     def4678
..
```

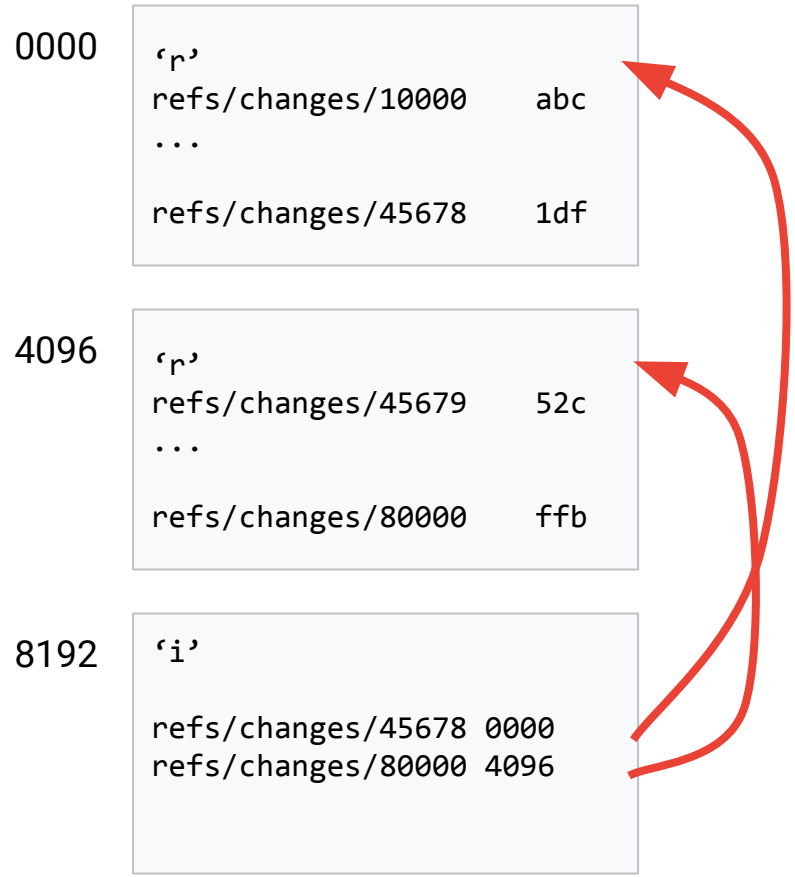
Blocks

- Storage likes to use blocks
  - 4kb default
  - 64kb at Google
- Store keys sorted
- Prefix compression for keys
- Index key restarts



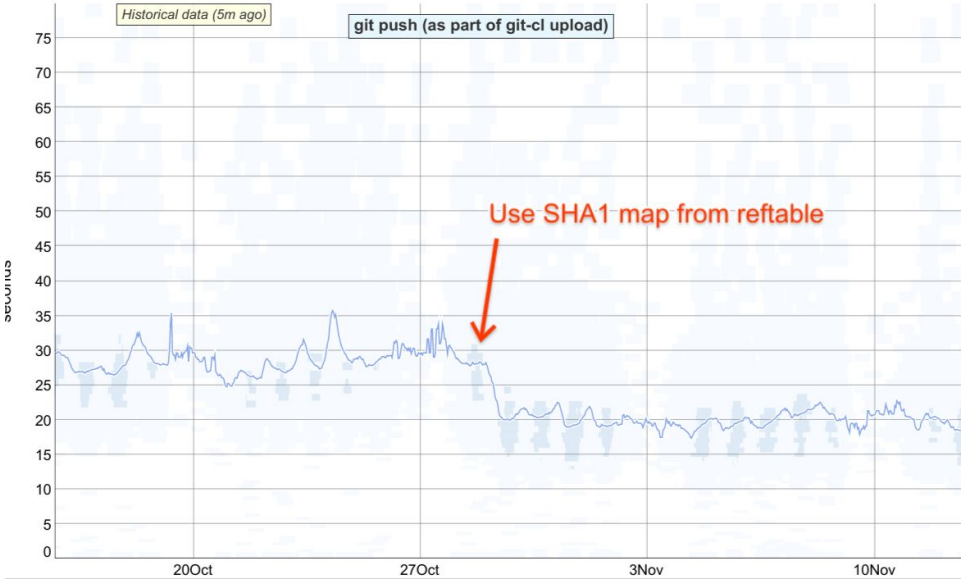
Block indexes

- Index minimizes number of seeks
- Index block holds
  - Key => Block offset
- Index blocks have same layout
  - Prefix compression!
- Large tables will have multiple levels



Fix other gripes too

- Store Reflog in “log” blocks
  - No more dir/file conflicts
  - RefDB + Reflog updates atomically
- Store SHA1 => ref mapping
  - Fast inverse lookups
  - Needed for visibility checks
  - Needed for Gerrit patch upload



Chromium push performance

Ref storage in Bigtable at Google

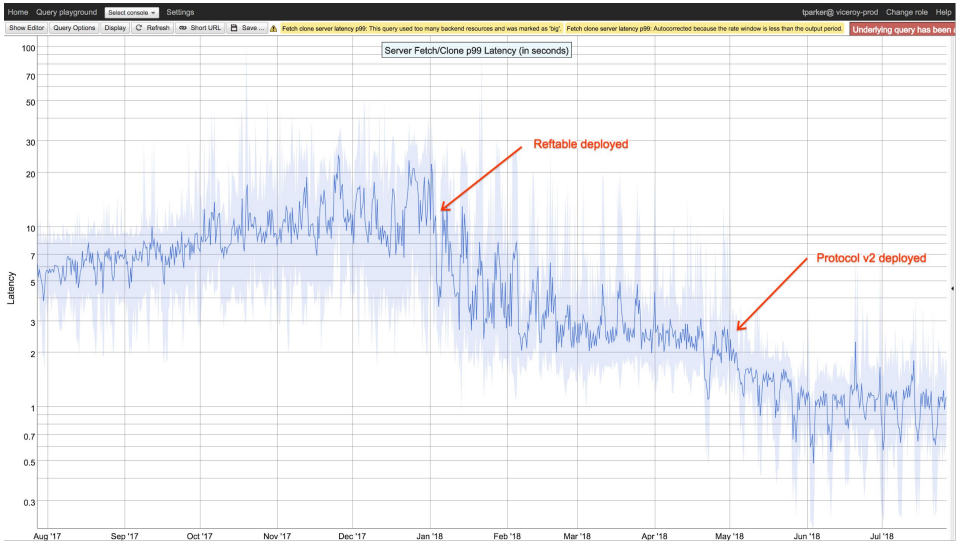
android:wifi:changes/1000	a1b2c3...
android:wifi:changes/1001	ff7dcb...
...	...

Encryption

android:NduoVksXjobBg	uoDCiroSd...
android:qX+uqdhzMCs7	9i37ZDCM...
...	...

Reftable history

- Aug 2017: Shawn Pearce introduces reftable format
- Dec 2017: reftable deployed at Google
- Nov 2019: JGit support in FileRepository



Demo

- Demo
- Measurements for write rate (synthetic)
  - 1ms/update (SSD, Linux/Mac)
  - 20 ms/update (NFS)
- Gerrit benchmark:
  - 1700 changes, SSD storage
  - Reftable
  - Packed-refs: 123ms / createchange (median)
  - Reftable: 71ms / createchange (median)

Outlook

- JGit: https://git.eclipse.org/r/c/146568/
- Library: https://github.com/google/reftable
  - Go - full implementation of (de)deserialization
  - C - the same; reflog storage missing
  - Plan: integrate into git-core
- CGit doesn’t support it yet
  - Hooks plugin?
- Ref storage is transparent to Gerrit
  - Go back and forth