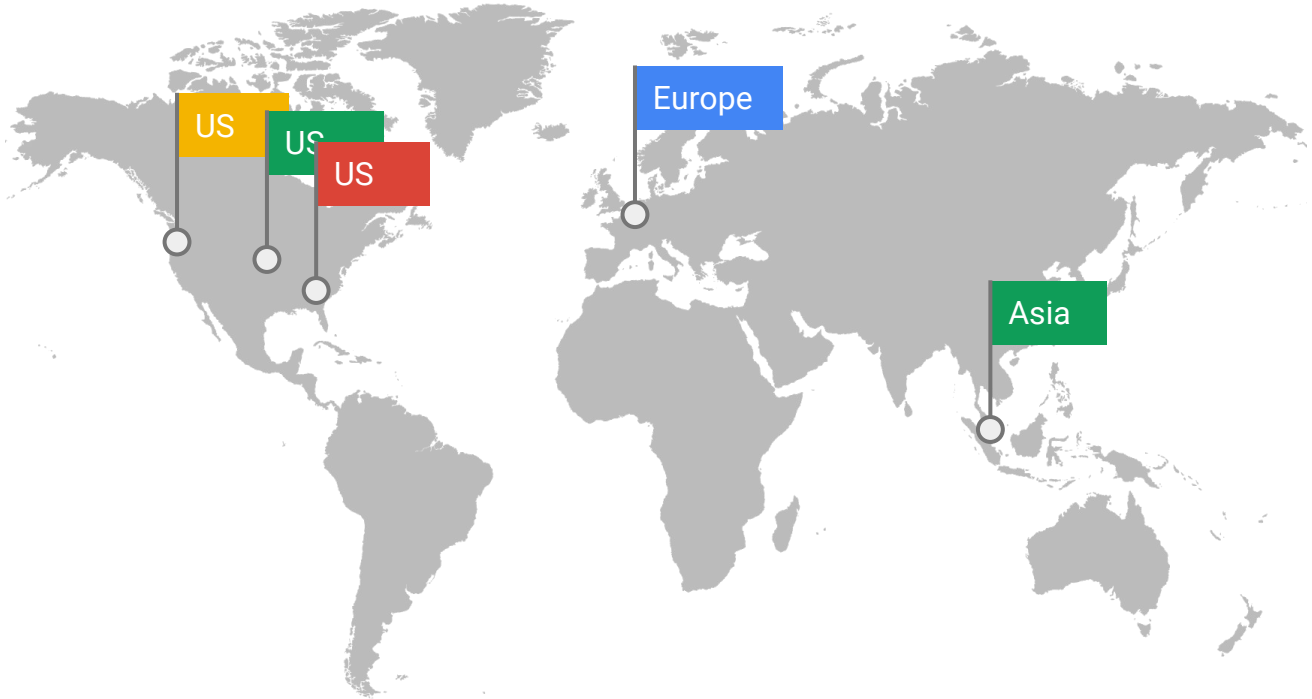
Google

Multi-master Gerrit

Shawn Pearce

Nov 8, 2015

Server locations



Gerrit at Google

Run from tip of master

and deliver to production ~every 2 days.
239 servers use same binary … and update together.

Database

is Google Megastore (global, 5.0 replication, Paxos).

Secondary index

is Google “Search Thingy” (global, eventual replication).
Conflicts resolved by change lastUpdatedOn timestamp.

* Small lie, we use a different servlet container and added some HTTP filters.

Only part of the story

Web sessions?

Caches inside Gerrit?

Git data?

Web sessions

Custom WebSession binding.
Pluggable in 2.10+.

Cookies from browser are stateless:

> Servers share a private key.
> Signs “Account.Id, timestamp” pair.

But, only half true.

Actually use same cookie system as GMail, etc. so that sign-out works across all Google products at once, including Gerrit.

2 hard problems in CS:

1. Cache invalidation
2. Naming things
3. Off by one errors

Disabled (some) caches

- accounts_byname
- changes
- change_kind

Forces database reads every time.

Great!

Database is replicated everywhere, consistently.
(Yay Paxos.)

No cache means reads are always current.
No cache means reads are slower(er).

```
$ cat gerrit.config

[cache "accounts_byname"]
    memoryLimit = 0

[cache "changes"]
    memoryLimit = 0

[cache "change_kind"]
    memoryLimit = 0
```

Immutable caches

Cache value depends on key (e.g. SHA-1 in key).

- conflicts
- diff
- diff_intraline
- git_tags
- permission_sort
- mergeability

Many are persisted to Google Bigtable.
Many are replicated using Bigtable replication.

Really like immutable caches.

No invalidation required.

Servers just read different key.
Try to load from Bigtable.
Recompute if missing.

Replication may save recompute.

Invalidate caches

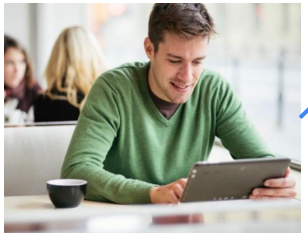
Hooked into Guava cache API.
Catch eviction calls made by Gerrit server.

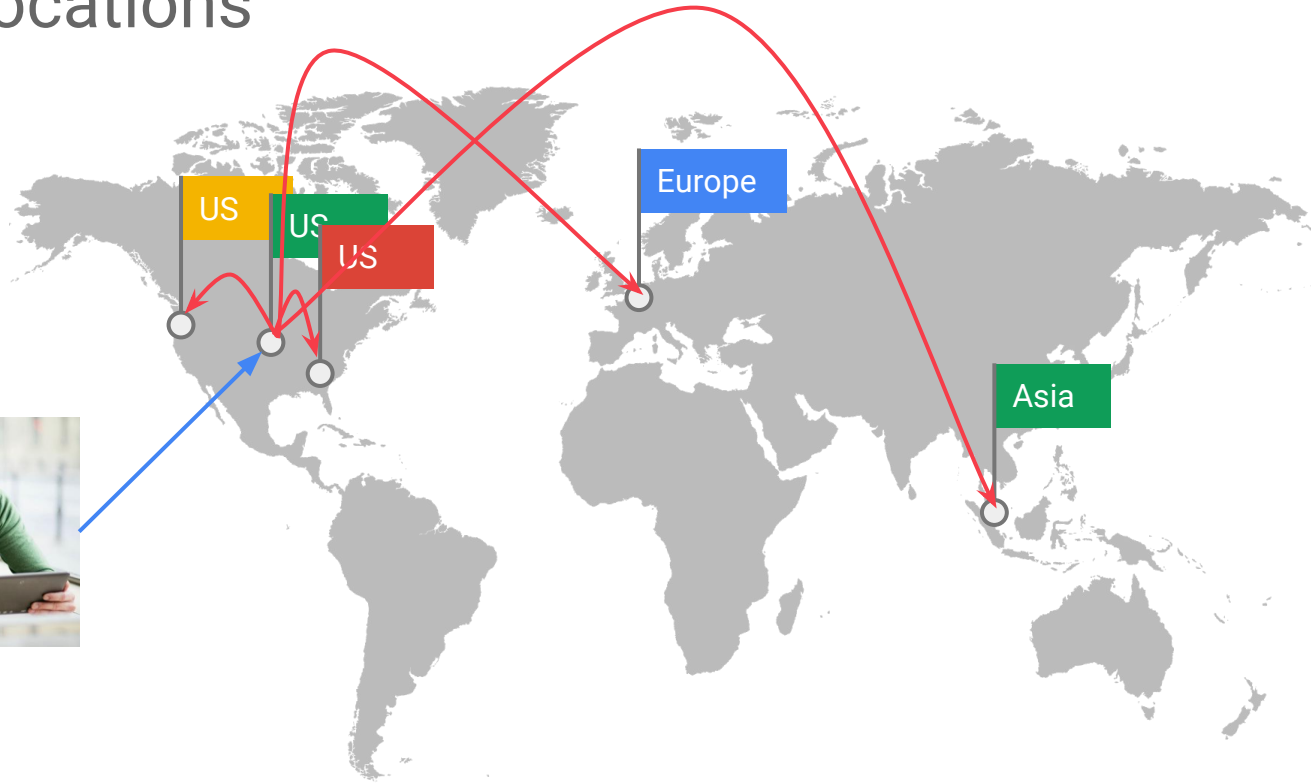
```
b.removalListener(new RemovalListener<>() {
    @Override
    public void onRemoval(RemovalNotification<> in) {
      if (in.getCause() == RemovalCause.EXPLICIT
          && PUBSUB_EVENT.get() != Boolean.TRUE) {
        msgQueue.send(in.getKey());
      }
    }
});
```

Forward key to peers using a message queue.
Must “de-bounce” with ThreadLocal<Boolean> PUBSUB_EVENT to prevent loops.

Git Paxos

Server locations





Scaling

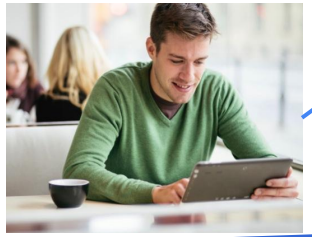
Partitioned capacity

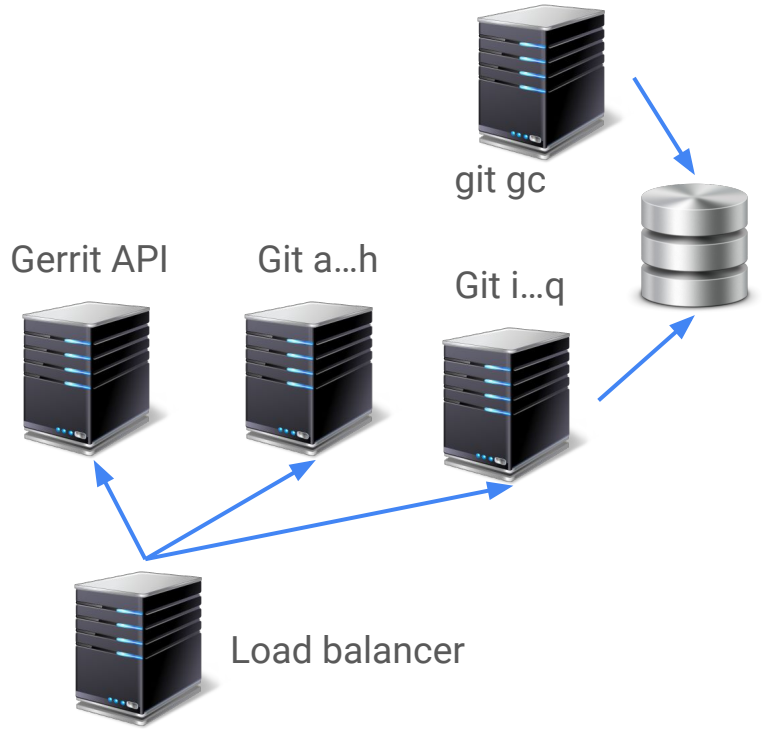
Segment responsibilities

- Gerrit REST API serving
- Git serving
- git gc

Shard repositories over multiple servers

- a…h on server0
- i…q on server1





Optimized garbage collection

Background
machine avoids CPU and RAM contention.

Push triggered
to repack frequently updated repositories.

“exproll”
to make gc alternately fast/slow.

> 500 ms
spent “Counting …” also triggers GC to rebuild bitmap indexes.

Future

Notedb

eliminates database, relies on Git Paxos.

Secondary index

updates per region after Git.

Sharding

may be required for REST API calls.

THANK YOU