Google

20 years of Bigtable

Lessons from running a large scale distributed system at Google

Gaurav Prabhu Gaonkar, John Tobin



Site Reliability Engineering

Agenda

Introduction



Background



Key Operational Challenges



Example Features We Have Built



Lessons Learned



Who are we?

- SREs at Google
- ~30 years on Bigtable, between the two of us
- John manages the Infra Bigtable SRE team in Dublin
- Gaurav is the overall TL of Infra and Cloud Bigtable SRE

Background

What is Bigtable?

A 3D sorted map

- A Bigtable *cell* holds many *tables* and self-hosts metadata
- A table contains lexicographically sorted rows with a key and many values, with a timestamp dimension
- Tables are broken into *tablets* with 1-N rows
- Tablets are hosted by *tablet servers*

For more info, search on Google for the "*Bigtable paper*" in OSDI and "*Twenty years of Bigtable*" in SIGMOD

Large Scale

- 7 Billion+ QPS
- 10 Exabytes+
- 1 Quadrillion+ (10^15) rows in the largest table
- 250M+ qps in a single cluster

Who uses Bigtable?

- Within Google:
  - Search, Ads, Youtube, Maps, ...
- Cloud Bigtable via GCP
- Low latency, high throughput workloads
- Machine learning, operational analytics, user facing applications

Early Days of Bigtable

- Each team was initially running their own Bigtable
- High adoption within Google
- Management overhead
- Lack of standardization
- Huge support burden
- Inefficient for Google

Motivation for Bigtable Service

- Managed multi-tenant service
- Establish best practices
- Help users solve problems, without requiring them to be Bigtable experts

Bigtable *Service* Development

- Bigtable Engineering created the core product, while Bigtable SRE built the control plane, self-service for users, automation, and SLOs to turn it into a reliable, managed *service*

Key Operational Challenges

Cattle, not pets

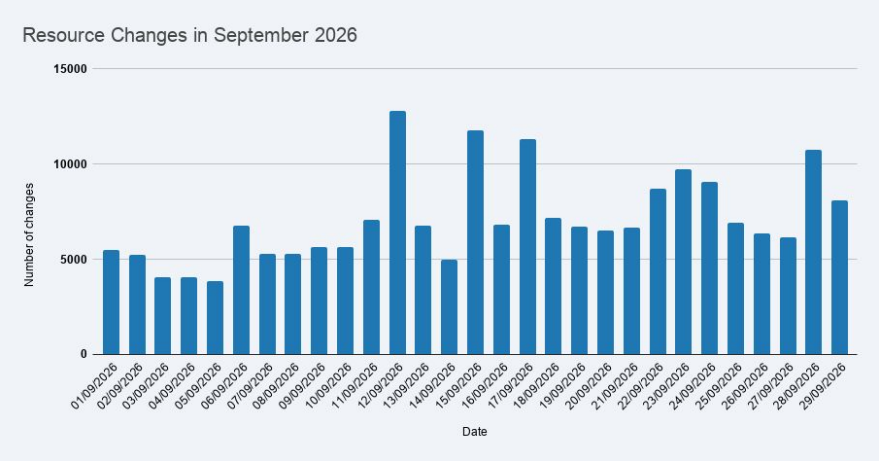
- Scaling from tens of Bigtable cells to thousands of Bigtable cells
- Support many different products without (much) per-product configuration
- Automated turnup and turndown

Isolation & Multi-tenancy

- Bigtable was originally designed as a single user system => no need for isolation
- Isolation is essential when anyone can start a 10,000 worker batch job on a shared Bigtable cell
- Quotas for each resource dimension
- Integrate with the quotas and isolation mechanisms provided by Colossus (distributed file system) and Borg (it inspired Kubernetes)

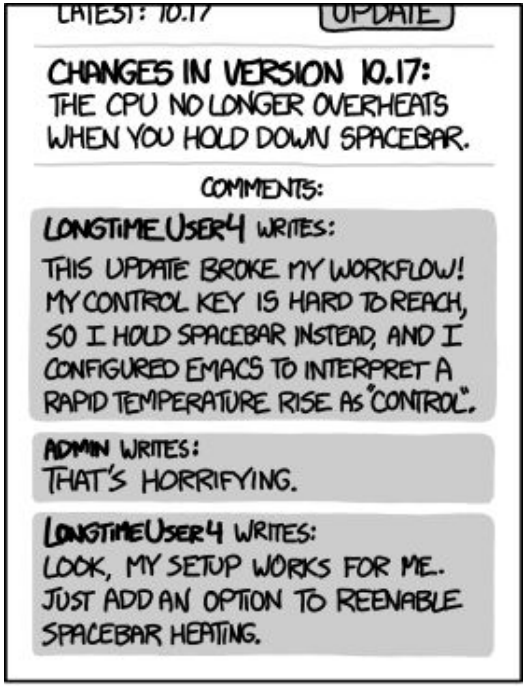
Self-service for users

- The graph on the right shows the number of resource changes per day in September 2026
- Reflects the scale and also the number of provisioning decisions being taken by automation



Hyrum's Law

- https://www.hyrumslaw.com
- Users depend on past behaviours, even when those are bad
- We sometimes have to maintain old behaviours for quarters or years for users to migrate their usage patterns



EVERY CHANGE BREAKS SOMEONE'S WORKFLOW.

Security

- Over time Bigtable stores more user data
- Bigtable SRE and Dev have no access to read or write user data
- Encryption at rest, encryption in transit
- We eagerly adopt all the security features Google develops



Features We Have Built

Isolation Improvements

- Initially: Single *partition* that hosted metadata and user data
- Separated metadata from user data
- User-specified separation of product workloads into different partitions
- Separation of background work and serving
- Automatically identify workloads that are triggering SLO violations and move them into dynamic *subpartitions*
- Cloud Bigtable users are always fully isolated

Resource management

- We can't centrally manage resources for all Bigtable users, so we made it self-service
- This includes turnups, turndowns, and changing server shapes

Autoscaling

- Manually increasing resources to handle organic growth is toilsome
- We built a user-configurable system to monitor resource usage and increase or decrease provisioning
- Organic growth and also inorganic demand
- This is built on top of the resource management system

Probers

- Synthetic monitoring / black box testing
- Probers for different paths
- Pre-Turnup Validation
  - Critical for testing new locations before user traffic moves in
- Meta-monitoring:
  - Monitoring the probers themselves to ensure alert integrity
- Prober deployment and cost tuning

Replication management

- Bigtable supports leader-follower and multi-primary replication
- Replication topologies are often complex and change for operational reasons
- We built the user-friendly system for managing and changing replication topologies

Replication monitoring

- Replication monitoring requires processing huge amounts of data across multiple clusters
- This does not fit well in the standard Google monitoring system
- We built a custom solution that became the basis for multiple Bigtable replication features

Service metadata

- Which Bigtable cells exist? What version should they be running?
- This and a lot more is stored in our service metadata

Lessons Learned

Reliability

- Reliability must be Feature #1
- Outages are inevitable
- Isolating failures to limit the blast radius
- Systemic Prevention: Address entire categories of problems instead of just the specific incidents

Efficiency

- Every % point increase in efficiency matters at scale
- Understand what is driving the costs
- Understand the tradeoffs
- Common patterns in efficiency
  - reducing headroom, reducing usage, avoiding wasted work
- Ranking different efficiency opportunities
- Validating the savings

Change Management

- Most of the outages are triggered by changes
- Every change should have metrics and monitoring
- Ensure your changes are validated properly
  - Non-trivial when you have a wide variety of workloads
- Progressive rollouts to reduce the blast radius of changes
  - Also applies to rollbacks!
- Visibility into traffic changes, dependency and environmental changes

Measure What Matters

- Know your CUJs
- Track User Happiness metrics, a customer should never have to tell you about an outage
- Documented SLIs, SLOs and SLAs
- Defend your SLOs with appropriate alerting
- Measure SLO compliance and review this regularly
- Integrate with new features and launches

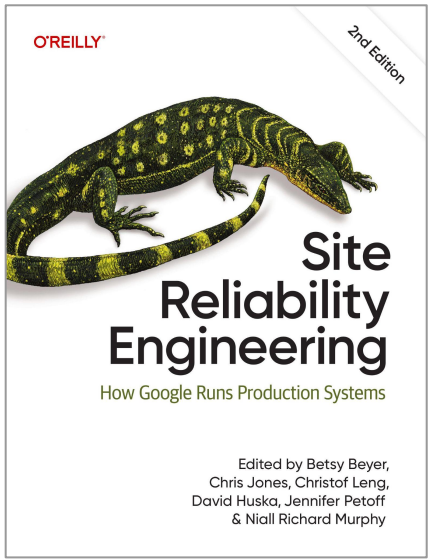
Long term health

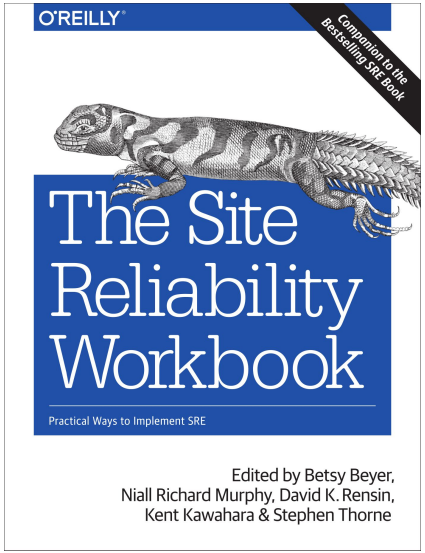
- Build and sustain good relationships with Bigtable Dev ***partners***
- Build and sustain good relationships with ***customers***, particularly those with unusual needs or scale
- Don't leave projects half-finished, address tech debt
- High depth of knowledge required - limited low hanging fruit
- Keep work interesting, motivate people

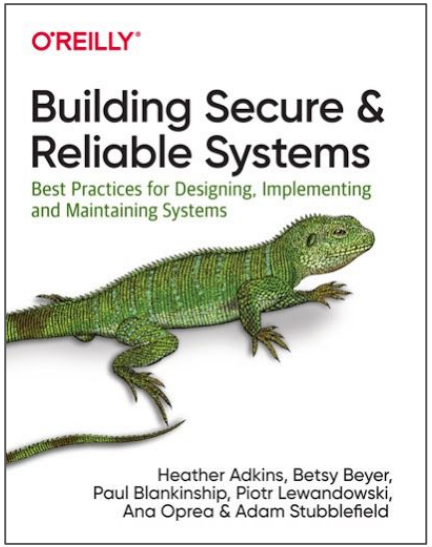
Team Culture

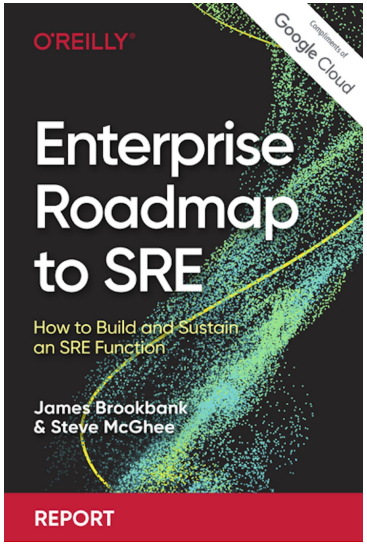
- Blameless postmortems
- No Heroes
- Working well in a distributed team: communication, documentation, avoiding knowledge silos
- Automate yourself out of your current job

Follow us on X: @googlesre. Find Google SRE publications—including the SRE Books, articles, trainings, and more—for free at sre.google/resources.









Book covers copyright O'Reilly Media. Used with permission.

Thank you

