Engineering Reliability into Web Sites: Google SRE

Alex Perry
alex.perry@google.com
Santa Monica SRE, Google

Google

Abstract

This talk introduces Site Reliability Engineering (SRE) at Google, explaining its purpose and describing the challenges it addresses.

SRE teams in Mountain View, Zürich, New York, Santa Monica, Dublin and Kirkland manage Google's many services and websites. They draw upon the Linux based computing resources that are distributed in data centers around the world.

Google

Outline

Dividing team responsibilities by site

Failures and instability consuming manpower

Engineering being applied to avoid future work

Migration of new projects into an SRE team

Planning for rapid growth in user community

Estimating the ideal size for an SRE team

Please ask directly relevant questions inline ...

Google

Site – an integrated deployment

Teams ensure user-visible uptime and quality

- Need authority over relevant software and systems

In depth knowledge of the details is necessary

- Steep learning curve, mostly due to complexity
- Continuous retraining, sites always being improved

Specializations for shared Grid infrastructure

- Ensure those components have good reliability

Google

Reliability – it just works

Responsible for minimizing manpower usage

- Team manages monitoring and develops automation
- Implies use of scripting and data analysis tools
- Most failures need automated recoveries in place

Elevated risk during convenient working hours

- Learn of age mortality risk during *preceding* workday
- Infant mortality ideally also avoids Google meals

Google

Engineering – not administration

Rigor in writing alert and notification definitions

- Holes may cause outage before notification occurs
- Routinely use multiple layers, levels and viewpoints
- Design the manual and automatic escalation paths

Responsible for enabling growth and scaling

- Plan for requirements, identify inefficiencies
- File bugs and, where appropriate, fix them too

Google

New idea, part time developer?

Sole contributor working one day per week

- Elsewhere, in the evenings or at the weekends
- Google engineers have 20% time for such projects

The developer probably has another idea too ...

- Quickly regrets how much work the old site needs!
- SREs have 20% time – is the project interesting?

Google

Initially, sites are high maintenance

SRE gives guidance in automating routine tasks

- Reduces workload by eliminating administrivia

SRE points out errors, omissions in documents

- Developer might then beg others for assistance

SRE suggests additional long term monitors

- These fill in coverage gaps and track performance
- Administrators need sufficient, trustworthy monitoring

Google

Launching a site is an opportunity

You shouldn't have to regret your new 20% site

- The pager may alert far more often than you'd like
  - (By default, site doesn't care about your working hours)

But, within weeks, SRE's expertise has its effect

- The developer's workload probably drops below 1%

The remaining 1% can be recovered any time

- Just write good docs so others can take over
- Many engineers choose to keep the pager instead

Google

Handing off a site to SRE

The decisions become progressively longer term

- Daily task workload for a site is getting reduced
- Software improvements are tuning and analysis

The developer still has a short term viewpoint

- Working on the next release, fixing known bugs
- The old live releases start to be a distraction
- An obvious incentive to request site transfer to SRE

Google

On call – more than quick fixes

SRE team members take turns in the rota

- Fix any problem whose solution is not yet automated
- Accumulate occurrence counts to identify priorities
- Document the effective diagnostics and solutions

The permanent solution takes a lot more time

- File bug, develop patch, test, code review, submit
- Schedule for integration, release and deployment
- Why spend many hours or days doing all that?

Google

Popularity – lots and lots of users

Site stability must not be impaired by growth

- Architecture will usually be scalable and robust
- These are key skills for Google software developers

Algorithm performance has a scale multiplier

- So do bugs, as well as monitoring and automation

Actual workload often scales with the issue itself

- Available manpower is limited to staff on hand ...

Google

O(): Scaling of issues – more users

Engineering issues may scale up with users “u”

- New features not yet implemented in software: O(1)
- Server down after routine hard drive failure: O(u)
- Cosmic ray in RAM, new user hash is valid: $O(u^2)$
- Bits flip in Ethernet, packet passes all CRC: $O(u^3)$

Higher order issues are initially very unlikely

- Effectively invisible and therefore cannot be fixed
- But their occurrence grows more rapidly with time

Google

First occurrence – to every hour

Imagine doubling users served every ten days

Consider an order $O(u^3)$ issue within the site

- instantaneous paging rate = exp (date / 5) / k
- cumulative count = exp (date / 5) 5 / k

After you receive the first page for that issue

- Expected first page date = 5 ln (k / 5)
- In three weeks, it would be paging every hour

Google

How fast can the site scale up?

This is primarily limited by issue resolution

- How long it takes to identify and then fix each one

Finding the bug, early, requires good tools

- Remote diagnostics and assertion logging
- Start developing solution before the second alert

Fixing the bug, only once, requires fast testing

- Ensure that old bugs don't come back again

Google

False positives, false negatives

Cautious alerts each become a crisis

- Lead to fixing immediate problems under pressure
- The opposite costs engineer time, and irritates

Analysis of archive data enables tuning

- Monitoring systems are routinely (correctly) silent
- New growing issues are then much easier to see
- Site can grow faster if issues are detected sooner

Google

Lower bound on SRE team size

1 day per week: Non-assigned 20% project work

1 day: Vacation, company events, volunteer, sick

1 day: Site specific training, learning changes

- Total per-site training cost is proportional to team size

2 days: Site analysis, planning, maintenance, etc

- How many man-days per week does the site require?
- This determines the lower bound on team size

Google

Estimating on call coverage needs

Can we risk engineers not responding to an alert?

- Probably not, so we need to implement redundancy

What is the failure rate of your paging services?

- Hopefully better than 10%, unlikely to achieve 1%
- eg: 5% with four way redundant paths is 99.999%

Only one engineer responds to any given alert

- Use a priority or election rule to avoid wasted effort
- The other SREs on call are unlikely to be disturbed

Google

Do sites need to be aggregated?

Compare site's engineering and on-call needs

- May place multiple sites into a single SRE team rota
- Training cost: Each engineer has to learn every site

This aggregation is always the long term goal

- At some point, every site is going to stop growing
- Afterward, engineers address low order issues
- For such a site, maintenance tends towards zero
- But the on call coverage requirement is constant

Google

Summary

- Site Reliability needs ongoing engineering effort
  - Longer timescale than software implementation
  - Working on both short and long deadlines is hard
- Large sites may have a team of administrators
  - If they're usually busy, there is no margin for growth
  - Sharing one team over many sites is cost effective
- Write / modify software instead of administering
  - Automation's benefits grow as the site scales up
- Teams should be small, each has an ideal size

Google

Thank you for your interest

Now, are there any questions ?

Or later?

alex.perry@google.com

Google