Striving for Imperfection

Using an error budget to move fast without compromising high reliability

Marc Alvidrez, SREcon15, March 16, 2015

Error budgets are one framework for managing risk

Can this work only at Google?

Session goal: Answer the question of how you can use error budgets in your environment

Let's have a conversation about error budgets

- Error budgets as a useful framework for managing risk [15m]
- Break into groups for discussion [20m]
- Report back [15m]
- Q&A [10m]

What is an error budget?

- Where did error budgets come from?
- How do they work?
- Why are they useful?

Photo credit: Xurble cc / cropped from original

First: A short journey through SLAs and metrics...

SLAs are a way of quantifying risk

Simplifying assumption:
SLAs are primarily about measuring unplanned downtime

SLA metrics: system uptime vs. request success rate

For availability SLAs we often talk about ***system uptime***:

$$availability = \frac{uptime}{(uptime + downtime)}$$

Another way to calculate this is with a ***request success rate***:

$$availability = \frac{successful\ requests}{total\ requests}$$

Defining SLAs in terms of request success rate makes it easier to define and measure an error budget

Where did error budgets come from?

AdSense circa 2006

- Major reimplementation of the AdSense serving system using technology from Search
- We saw the typical problems from a new large system
  - Instability, many pages...

We found it challenging to hit our SLAs

Google AdSense

Exceeding our SLA meant we had an opportunity

AdSense circa 2009

Google AdSense

- The serving system was stable, efficient and performs well
- **Consistently exceeding our SLA targets**
- Perhaps it performs a little ***too*** well?

Instead of exceeding our SLA we could have...

- Been moving faster
  - Push code more often or with less QA
  - Roll out system configuration changes more quickly
- Improved the quality of life for our engineers
  - Lower page levels, higher response times
- Been running leaner
  - Run hotter with less hardware

We could have been taking more risks!

SLAs are minimum *and* maximum targets

SRE's goal is not to make systems that are 100% reliable. Our goal is to engineer systems that are only ***as reliable as necessary***.

Hang on: Aren't all failures bad?

Usually, no: SLAs are a cost benefit exercise

We started trying to find ways to take more, thoughtful risks

What does it mean to take more thoughtful risks?

Going back to the example of AdSense, in 2009 we built a ***1% cluster***

- It was a serving cluster sized to handle 1% of our traffic at peak
- We ran it like any other cluster: the same monitoring, alerting, etc.
- We used it to move faster:
  - We set up automation to push new builds of code there on a daily basis
  - We used it as the place to verify new configuration
- What happens if it breaks?
  - We get paged, we react as usual, and failover that (up to) 1% of traffic

The error budget is the headroom you have above your SLA

Error budgets present a ***quantitative approach to managing risk.***

$$availability = \frac{successful\ requests}{total\ requests}$$

Using the ***request success rate*** to measure availability, we can define an ***error budget*** like this:

$$error\ budget = availability - SLA\ target$$

This framing is useful because it allows us to take ***engineering approach*** to managing service risk: measure → analyze → improve

Error budgets turn out to be generically useful

You can use them for ***many types of systems***:

- Infrastructure
- Low latency serving
- Batch oriented pipelines

The SLA metrics can differ from system to system, but the framework holds

An error budget is also useful for ***managing engineering investment*** (i.e. a metric to use to avoid "gilding the lily")

What are the minimum requirements for using an error budget?

1. You need to know your ***SLA threshold***
2. You need to be able to measure your ***availability performance***
3. Your availability performance needs to ***consistently exceed*** your SLA threshold

Is there an easy and concrete way to get started?

Recommendation: Choose a reasonable but arbitrary metric and threshold to get started

Example: An interactive web application served over HTTP to end users across the Internet

SLA threshold: 99.9% external, 99.95% internal
Errors: 500s served by your web server
Measurement method: simple script to grab web server logs daily, count successes and failures, and calculate your availability

Error budget framework summary

- Error budgets are a tool for managing risk
- It is the performance headroom above your SLA
- At a minimum: you need to know your SLA threshold, measure your performance, and consistently
- They are one way to unlock SRE's ability to take thoughtful risks

Agenda

- Error budgets as a useful framework for managing risk [15m]
- Break into groups for discussion [20m]
- Report back [15m]
- Q&A [10m]

Questions

- What is the project you want to talk about?
- Do you have a measure (for availability) that you want to optimize around?
- Is there a tradeoff you can make in exchange for lowering that measure?
- What room is there for play between what you can achieve for availability, and what is acceptable to the users of the project?
- What supporting structures or process do you need to run and maintain the error budget?

Report back

Q&A

Google

Thank you

Marc Alvidrez <cram@google.com>

Additional Slides

How do you set an SLA?

The right availability SLA is a judgement call:

- Talk to the ***product/service owners*** to figure out the right availability target for a service

Questions to ask:

- What and who is the service for? What are the expectations? Does it have to be fast? Can it be down sometimes? Does it tie to revenue? Who is the target customer? Is it a free or paid service?

Where to start:

- Set an ***arbitrary SLA threshold*** to focus the conversation
- Set an ***arbitrary metric to measure*** to establish a starting point

SLA Measurement

How do you measure failures?

- Monitoring data
- Log analysis
- Sampling

Key considerations:

- Not all requests are equal
- Quality vs. availability monitoring
- Pipelines: throughput delay, deadlines
- Infrastructure: higher targets, higher cost, greater potential reward

What about when you're not hitting your SLA?

Tactical responses:

- Freeze changes
- Partition workloads

Re-engineer for:

- Overload protection
- Cascading failure resistance
- Graceful degradation
- Dynamic stability

Spending your error budget

Things you might want to buy:

- Move faster (e.g. higher rate of change, new features, speed to market, innovate)
- Run tighter (e.g. build less redundant capacity)
- Better quality of life

Key considerations:

- Bound the risk you're exposing yourself to
- Choose risks that are easily reversible (e.g. easy failover)
- Choose risks that are easier to measure

Goal: Thoughtful, bounded risk taking