

How to Include Latency in SLO-based Alerting

Björn “Beorn” Rabenstein

KubeCon + CloudNativeCon NA, San Diego – 2019-11-20



Act 1: What this talk doesn't cover

SLI/SLO/SLA – words momentous calmly hast thou spoken...

Service Level Objectives

Written by Chris Jones, John Wilkes, and Niall Murphy with Cody Smith

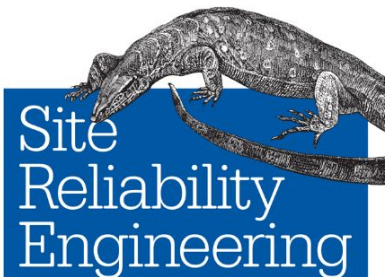
Edited by Betsy Beyer

It's impossible to manage a service correctly, let alone well, without understanding which behaviors really matter for that service and how to measure and evaluate those behaviors. To this end, we would like to define and deliver a given **level of service** to our users, whether they use an internal API or a public product.

We use intuition, experience, and an understanding of what users want to define **service level indicators** (SLIs), **objectives** (SLOs), and **agreements** (SLAs). These measurements describe basic properties of metrics that matter, what values we want those metrics to have, and how we'll react if we can't provide the expected service. Ultimately, choosing appropriate metrics helps to drive the right action if something goes wrong, and also gives an SRE team confidence that a service is healthy.

This chapter describes the framework we use to wrestle with the problems of metric modeling, metric selection, and metric analysis. Much of this explanation would be quite abstract without an example, so we'll use the Shakespeare service outlined in [Shakespeare: A Sample Service](#) to illustrate our main points.

O'REILLY



HOW GOOGLE RUNS PRODUCTION SYSTEMS

<https://www.usenix.org/conference/srecon19emea/presentation/desai>

The Map Is Not the Territory: How SLOs Lead Us Astray, and What We Can Do about It

Thursday, 2019, October 3 - 09:45-10:30
Narayan Desai, Google

Abstract:

SLOs are a wonderfully intuitive concept: a quantitative contract that describes expected service behavior. These are often used in order to build feedback loops that prioritize reliability, communicate expected behavior when taking on a new dependency, and synchronize priorities across teams with specialized responsibilities when problems occur, among other use cases. However, SLOs are built on an implicit model of service behavior, with a raft of simplifying assumptions that don't universally hold.

These simplifying assumptions make SLO rules of thumb fall apart with complex modern services, which can result in bad decision making. In this talk, I will catalog a range of these issues with SLOs and demonstrate how they cause systematic failures of SLO-based processes. Armed with the knowledge of these failure modes, I'll present a set of best practices for understanding when SLOs produce incorrect and unexpected results and a set of techniques for constructing robust SLOs.

Narayan Desai is an SRE at Google, where he focuses on the reliability of Google Cloud Platform Data Analytics products. He has a checkered past, having worked on scheduling, configuration management, supercomputers, and metagenomics—always in the context of production systems.

Twitter: [@nldesai](#)

Latency SLOs Done Right

Wednesday, 2019, October 2 - 11:30-12:00

Heinrich Hartmann, Circonus

Abstract:

Latency is a key indicator of service quality, and important to measure and track. However, measuring latency correctly is not easy. In contrast to familiar metrics like CPU utilization or request counts, the "latency" of a service is not easily expressed in numbers. Percentile metrics have become a popular means to measure the request latency, but have several shortcomings, especially when it comes to aggregation. The situation is particularly dire if we want to use them to specify Service Level Objectives (SLOs) that quantify the performance over a longer time horizons. In the talk we will explain these pitfalls, and suggest three practical methods how to implement effective Latency SLOs.



Heinrich Hartmann is the Analytics Lead at Circonus. He is driving the development of analytics methods that transform monitoring data into actionable information as part of the Circonus monitoring platform. In his prior life, Heinrich pursued an academic career as a mathematician. Later he transitioned into computer science and worked as a consultant for a number of different companies and research institutions.

Twitter: [@HeinrichHartman](https://twitter.com/HeinrichHartman)



Alerting on SLO breaches

vs.

Billing on SLA Breaches

Monitoring Distributed Systems

Written by Rob Ewaschuk

Edited by Betsy Beyer

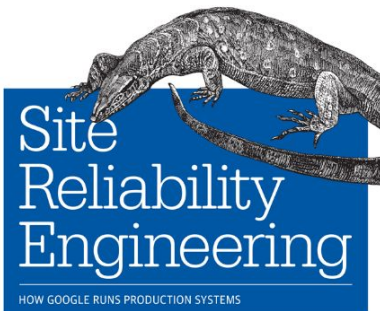
Google's SRE teams have some basic principles and best practices for building successful monitoring and alerting systems. This chapter offers guidelines for what issues should interrupt a human via a page, and how to deal with issues that aren't serious enough to trigger a page.

Definitions

There's no uniformly shared vocabulary for discussing all topics related to monitoring. Even within Google, usage of following terms varies, but the most common interpretations are listed here.

Monitoring

O'REILLY



Practical Alerting from Time-Series Data

Written by Jamie Wilkinson

Edited by Kavita Guliani

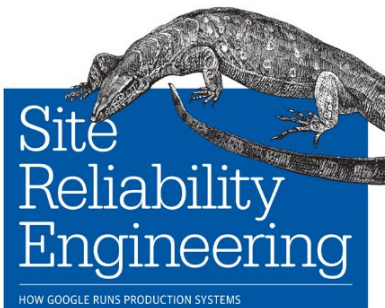
“ *May the queries flow, and the pager stay silent.* ”

Traditional SRE blessing

Monitoring, the bottom layer of the **Hierarchy of Production Needs**, is fundamental to running a stable service. Monitoring enables service owners to make rational decisions about the impact of changes to the service, apply the scientific method to incident response, and of course ensure their reason for existence: to measure the service's alignment with business goals (see [Monitoring Distributed Systems](#)).

Regardless of whether or not a service enjoys SRE support, it should be run in a symbiotic relationship with its monitoring. But having been tasked with ultimate responsibility for Google Production, SREs develop a particularly intimate knowledge of the monitoring infrastructure that supports their service.

O'REILLY



Alerting on SLOs

By Steven Thurgood

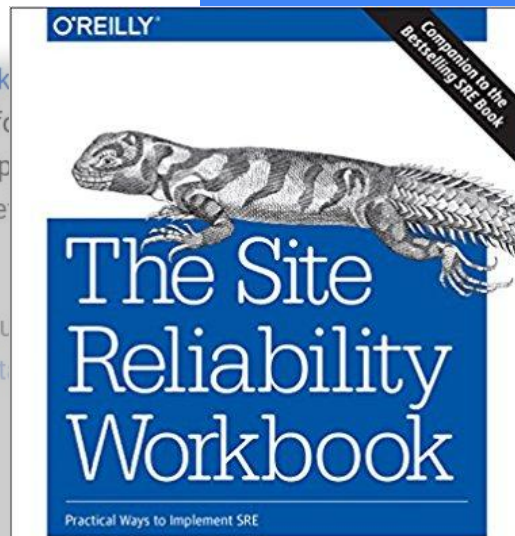
with Jess Frame, Anthony Lenton,

Carmela Quinito, Anton Tolchanov, and Nejc Trdin

This chapter explains how to turn your SLOs into actionable alerts on significant events. Both our [first SRE book](#) and [second SRE book](#) talk about implementing SLOs. We believe that having good SLOs that measure the reliability of your platform as experienced by your customers, provides the highest-quality indication for when an on-call engineer should respond. In this chapter, we give specific guidance on how to turn those SLOs into alerting rules so that you can respond to problems before they consume too much of your error budget.

Our examples present a series of increasingly complex implementations for alerting metrics and logic; we discuss the utility and shortcomings of each. While our examples use a simple request-driven service and [Prometheus syntax](#), you can apply this approach in any alerting framework.

Alerting Considerations



Backstage Blog June 4th, 2019 [Monitoring](#) [Prometheus](#) [SRE](#)

Alerting on SLOs like Pros

By Björn "Beorn" Rabenstein

If there is anything like a silver bullet for creating meaningful and actionable alerts with a high signal-to-noise ratio, it is alerting based on service-level objectives (SLOs). Fulfilling a well-defined SLO is the very definition of meeting your users' expectations. Conversely, a certain level of service errors is OK as long as you stay within the SLO – in other words, if the SLO grants you an *error budget*. Burning through this error budget too quickly is the ultimate signal that some rectifying action is needed. The faster the budget is burned, the more urgent it is that engineers get involved.

This post describes how we implemented this concept at SoundCloud, enabling us to fulfill our SLOs without flooding our engineers on call with an unsustainable amount of pages.

Let's Talk SRE

<https://developers.soundcloud.com/blog/alerting-on-slos>

Developers

Your Apps

~~Register a new app~~
(Currently unavailable)

HTTP API Guide

HTTP API Reference

HTTP API Wrappers

Widget API

Rate Limits

Support

<https://youtu.be/gg04hEoQsl4>



0:17 / 48:33 [Play Icon] [Next Icon] [Volume Icon] [CC Icon] [Subtitles Icon] [Fullscreen Icon]

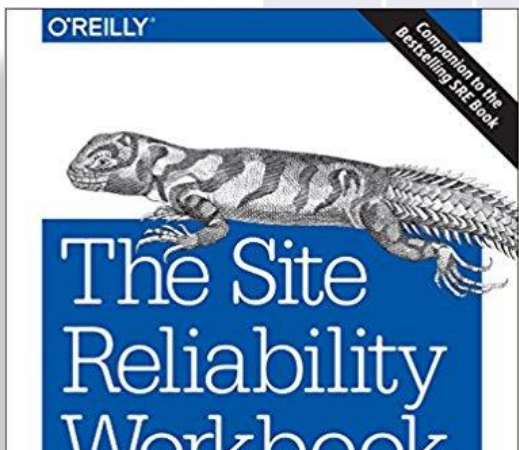
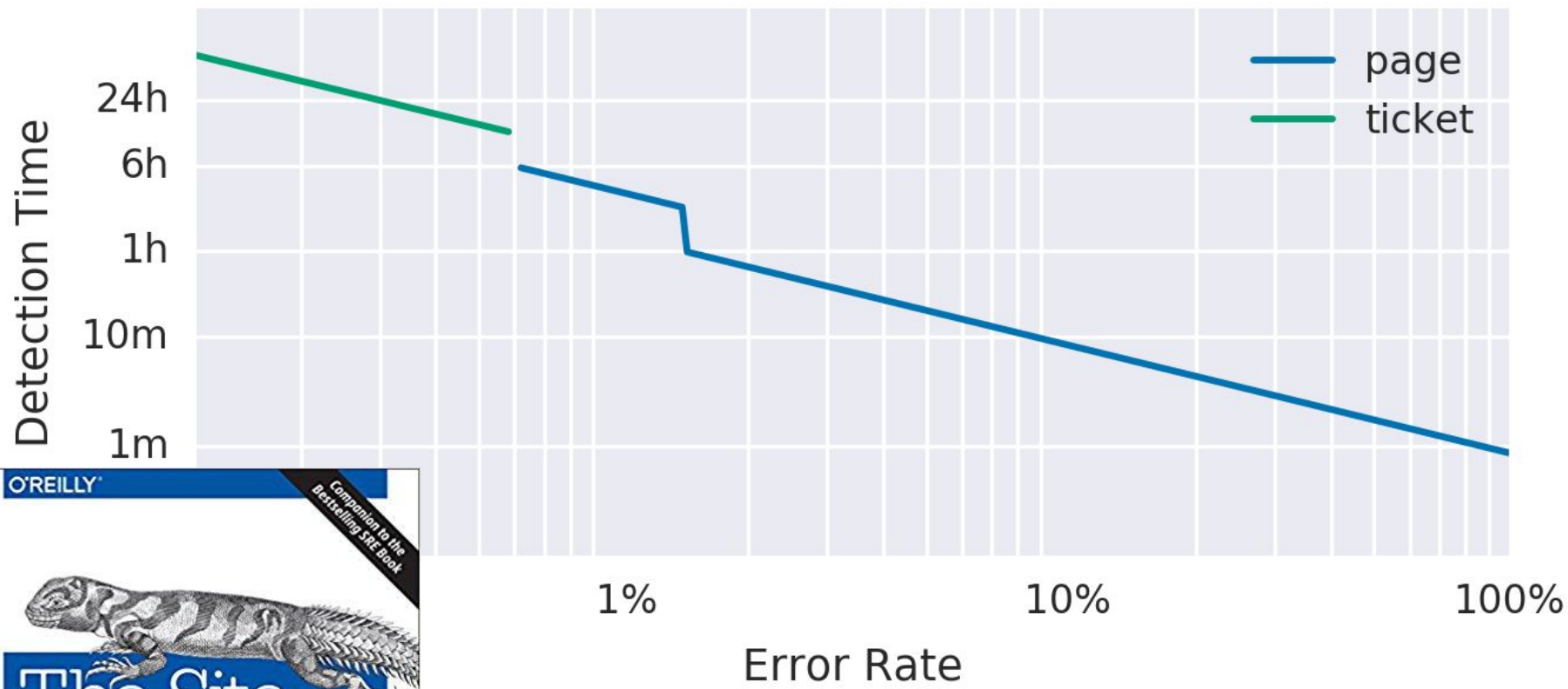
Alerting on SLOs like Pros Up next **AUTOPLAY**

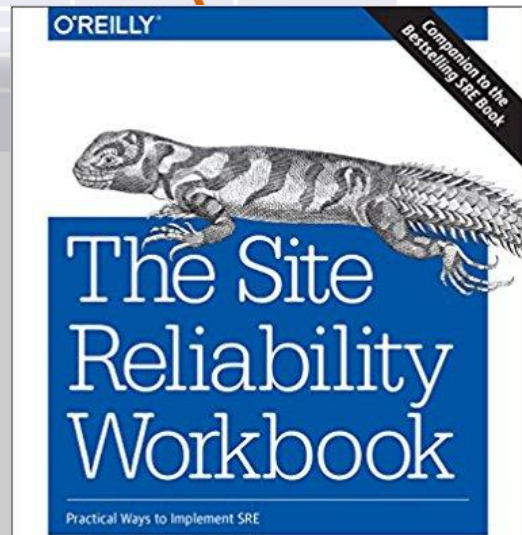
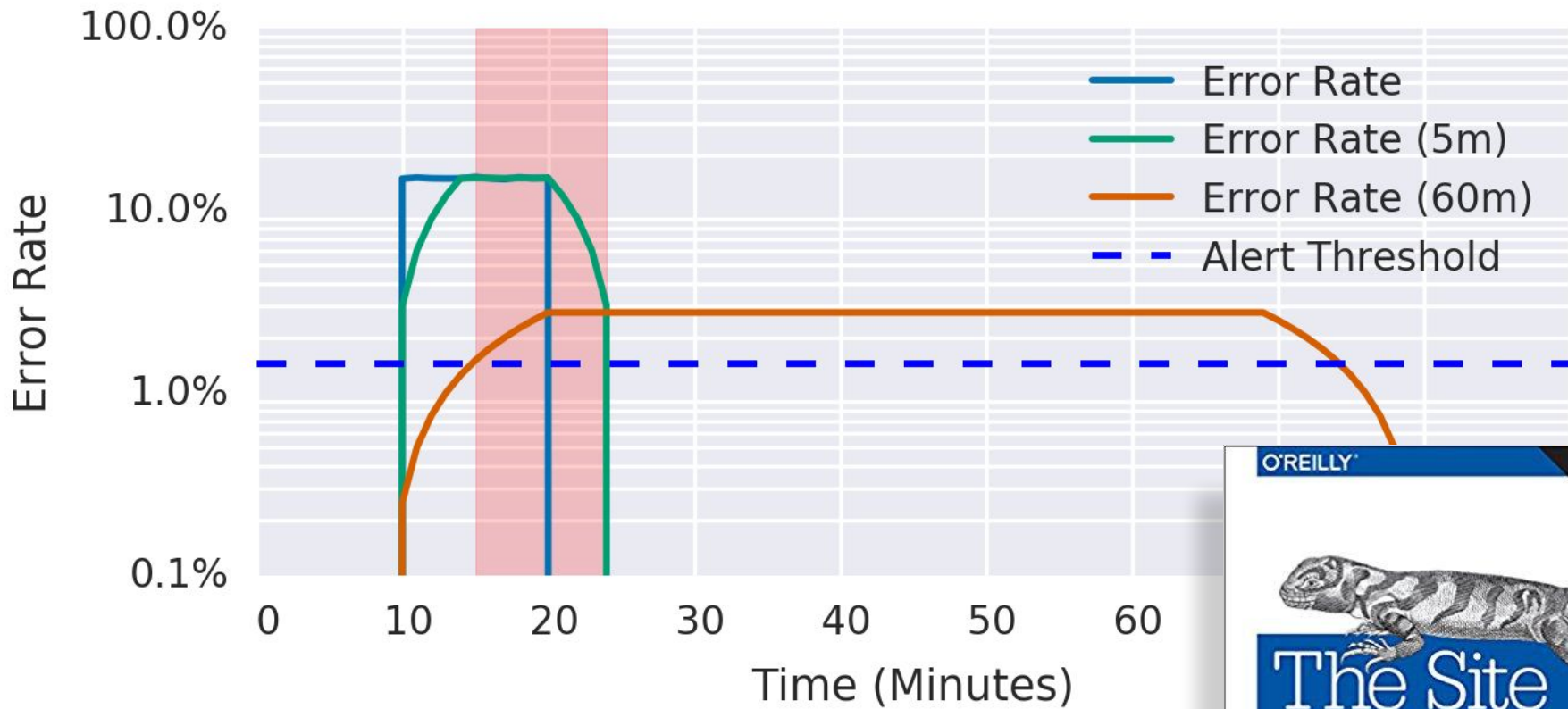
105 views · Sep 27, 2019 👍 1 👎 0 ➦ SHARE 📌 SAVE ⋮

Visual Basic .Net : Search in Access Database - ...

Alert	Long window	Short window	for duration	Burn rate factor	Error budget consumed
Page	1h	5m	2m	14.4	2%
Page	6h	30m	15m	6	5%
Ticket	1d	2h	1h	3	10%
Ticket	3d	6h	1h	1	10%







Alert	Long window	Short window	for duration	Burn rate factor	Error budget consumed
Page	1h	5m	2m	14.4	2%
Page	6h	30m	15m	6	5%
Ticket	1d	2h	1h	3	10%
Ticket	3d	6h	1h	1	10%

```
// According to https://developers.soundcloud.com/blog/alerting-on-slos :  
local windows = [  
  { long_period: '1h', short_period: '5m', for_period: '2m', factor: 14.4, severity: 'critical' },  
  { long_period: '6h', short_period: '30m', for_period: '15m', factor: 6, severity: 'critical' },  
  { long_period: '1d', short_period: '2h', for_period: '1h', factor: 3, severity: 'warning' },  
  { long_period: '3d', short_period: '6h', for_period: '3h', factor: 1, severity: 'warning' },  
];
```

```
- alert: AmpelmannErrorBudgetBurn
  expr: |2
    (
      100 * backend:http_errors_per_response:ratio_rate1h
    > on (backend)
      14.4 * backend:error_slo:percent
    )
  and
    (
      100 * backend:http_errors_per_response:ratio_rate5m
    > on (backend)
      14.4 * backend:error_slo:percent
    )
  for: 2m
  labels:
    system: "{{ $labels.backend }}"
    severity: "critical"
    window: "1h"
  annotations:
    summary: "a backend burns its error budget very fast"
    description: "Backend {{ $labels.backend }} has returned {{ $value | printf `%.2f` }}% 5xx"
    runbook: "http://runbooks.soundcloud.com/runbooks/ampelmann/#ampelmannerrorbudgetburn"
```




Matthias Loibl
metalmatze

Unfollow

★ PRO

Software Engineer working on monitoring with Prometheus and Kubernetes at Red Hat CoreOS. Interested in web development, distributed systems and metal.

Red Hat

Berlin

mail@matthiasloibl.com

http://matthiasloibl.com

Block or report user

Organizations



<https://github.com/metalmatze/slo-libsonnet>

<https://github.com/metalmatze/slo-libsonnet-web>

<https://promtools.matthiasloibl.com/>

PromTools

SLOs with Prometheus

Multiple Burn Rate Alerts

This page will generate, with the data you provide in the form, the necessary Prometheus alerting and recording rules for [Multiple Burn Rate](#) which you might know from [The Site Reliability Workbook](#). These rules will evaluate based on the available metrics in the last **30 days**.

Availability (Unavailability in 30d: 43min)

99.9

Availability is generally calculated based on how long a service was unavailable over some period.

Metric

http_requests_total

The metric name to base the SLO on.

It's best to base this on metrics coming from a LoadBalancer or Ingress.

Name

job

Value

prometheus

Generate


+

Act 2: Musing about latency SLOs


Or even SLAs...

The background is a dark blue gradient with several abstract geometric elements. There are thin, light blue curved lines that sweep across the frame. Scattered throughout are various small shapes: a cluster of four blue squares in the upper left, a single green dot in the upper center, a cluster of four green dots in the upper right, a small blue triangle in the lower left, and a small orange bar chart icon inside a white circle in the lower right. The overall aesthetic is clean, modern, and tech-oriented.


“We guarantee an uptime of 99.9%.”

The background is a dark blue gradient with several light blue circular arcs and lines. There are several small icons: a cluster of four blue squares in the top left, a single green dot in a dark blue circle in the top center, a cluster of five green dots in the top right, a small blue triangle in the bottom left, and a bar chart icon in a white circle in the bottom right.

“During each month, we’ll serve 99.9% of requests successfully.”



“During each month, we’ll serve 99.9% of requests successfully. The 99th percentile latency will be below 500ms.”



“During each month, we’ll serve 99.9% of requests successfully within 500ms.”

<https://www.usenix.org/conference/srecon19emea/presentation/fouquet>

SLOs for Data-Intensive Services

Wednesday, 2019, October 2 - 11:00-11:30

Yoann Fouquet, Booking.com

Abstract:

Designing and maintaining a search engine service can be challenging. One of the challenges is to set insightful SLOs where standard availability/latency SLOs do not fit. We will go through our journey towards defining a monitoring process for such services at Booking.com, from ineffective availability/latency SLOs to the current setup and all its advantages; travelling in a world where providing accurate and consistent responses can be as important as availability.



Yoann is a Site Reliability Engineer at Booking.com, working on core services within the Booking.com infrastructure.

Open Access Media

USENIX is committed to Open Access to the research presented at our events. Papers and proceedings are freely available to everyone once the event begins. Any video, audio, and/or slides that are posted after the event are also free and open to everyone. **Support USENIX** and our commitment to Open Access.

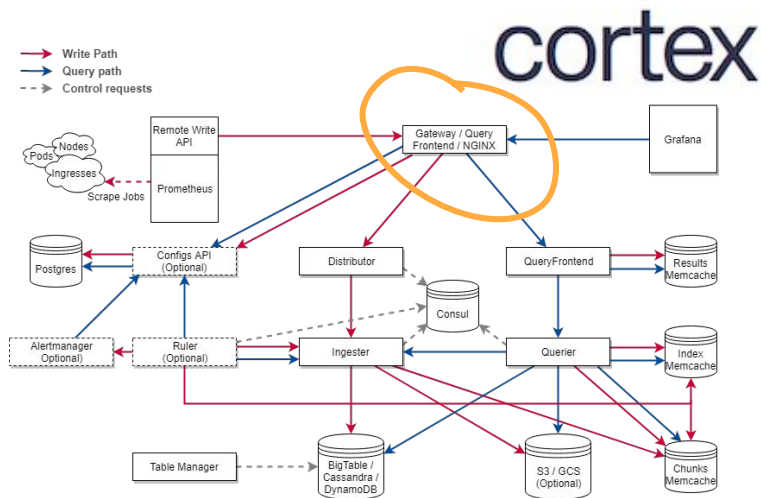
Act 3: A pragmatic implementation

Ingredient: a partitioned histogram at the edge



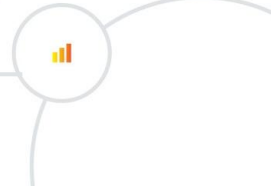
Prometheus Histogram on Cortex GW
cortex_request_duration_seconds
with buckets for 1s and 2.5s (and more!)
partitioned(!) by

- status_code
- method
- route



BTW: Histograms to become cheaper. Shameless plug:

<https://promcon.io/2019-munich/talks/prometheus-histograms-past-present-and-future/>



SLO

Complete 99.9% of writes successfully in less than 1s.
Respond to 99.5% of reads in less than 2.5s.



```
record: namespace_job:cortex_gateway_write_slo_errors_per_request:ratio_rate1h
```

```
expr: |2
```

```
1 -  
(  
  sum by(namespace, job) (  
    rate(cortex_request_duration_seconds_bucket{le="1",route="push",status_code!~"5.."}[1h])  
  )  
/  
  sum by(namespace, job) (  
    rate(cortex_request_duration_seconds_count{route="push"}[1h])  
  )  
)
```

```
record: namespace_job:cortex_gateway_read_slo_errors_per_request:ratio_rate1h
```

```
expr: |2
```

```
1 -  
(  
  sum by(namespace, job) (  
    rate(cortex_request_duration_seconds_bucket{le="2.5",route="query",status_code!~"5.."}[1h])  
  )  
/  
  sum by(namespace, job) (  
    rate(cortex_request_duration_seconds_count{route="query"}[1h])  
  )  
)
```

Alert	Long window	Short window	for duration	Burn rate factor	Error budget consumed
Page	1h	5m	2m	14.4	2%
Page	6h	30m	15m	6	5%
Ticket	1d	2h	1h	3	10%
Ticket	3d	6h	1h	1	10%

```
// According to https://developers.soundcloud.com/blog/alerting-on-slos :  
local windows = [  
  { long_period: '1h', short_period: '5m', for_period: '2m', factor: 14.4, severity: 'critical' },  
  { long_period: '6h', short_period: '30m', for_period: '15m', factor: 6, severity: 'critical' },  
  { long_period: '1d', short_period: '2h', for_period: '1h', factor: 3, severity: 'warning' },  
  { long_period: '3d', short_period: '6h', for_period: '3h', factor: 1, severity: 'warning' },  
];
```

```
alert: CortexWriteErrorBudgetBurn
expr: |2
  (
    100 * namespace_job:cortex_gateway_write_slo_errors_per_request:ratio_rate1h
    > 0.1 * 14.4
  )
and
  (
    100 * namespace_job:cortex_gateway_write_slo_errors_per_request:ratio_rate5m
    > 0.1 * 14.4
  )
for: 2m
labels:
  period: 1h
  severity: critical
annotations:
  description: '{{ $value | printf `%.2f` }}% of {{ $labels.job }}'s write requests
    in the last 1h are failing or too slow to meet the SLO.'
  summary: Cortex burns its write error budget too fast.
```

```
alert: CortexReadErrorBudgetBurn
expr: |2
  (
    100 * namespace_job:cortex_gateway_read_slo_errors_per_request:ratio_rate1h
    > 0.5 * 14.4
  )
and
  (
    100 * namespace_job:cortex_gateway_read_slo_errors_per_request:ratio_rate5m
    > 0.5 * 14.4
  )
for: 2m
labels:
  period: 1h
  severity: critical
annotations:
  description: '{{ $value | printf `%.2f` }}% of {{ $labels.job }}'s read requests
    in the last 1h are failing or too slow to meet the SLO.'
  summary: Cortex burns its read error budget too fast.
```

Conclusions

1. Meaningful latency SLAs are actually quite relevant.
2. SLO-based alerting is a great idea.
3. Include latency into your SLO-based alerting.
4. Which could very well evolve into a real latency SLA.
5. Keep it as simple as possible.
6. But not simpler.





<https://github.com/beorn7/talks>

beorn@grafana.com