

Latency SLOs Done Right

Heinrich Hartmann

FOSDEM 2019, Brussels

@heinrichhartmann

For any API you care about ...

How many requests in January were served within 100ms?

For any API you care about ...

How many requests in January were served within **150ms**?

For any API you care about ...

How many requests in January were served within **180ms**?

For any API you care about ...

How many requests in **June 16th 9:12-9:35** were served within 100ms?



Hi, I am Heinrich

- Data Scientist at Circonus
- PhD in Mathematics
- Talks about #StatisticsForEngineers
- Lives in Stemwede, Germany
- @HeinrichHartmann on Twitter



@heinrichhartmann

Agenda

- Why monitor Latency?
- What is an SLO?
- Three methods to calculate Latency SLOs
 - Method 1
 - Method 2
 - Method 3
- Conclusion

What is an SLO?

Service Level Indicator (SLI)

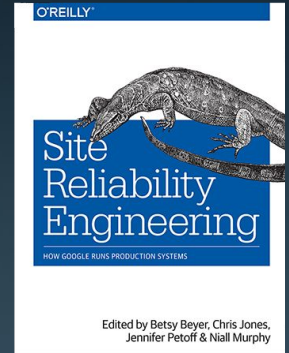
A metric that quantifies the quality or reliability of your service.

Service Level Objective (SLO)

A target value for the service level, as measured by an SLI, that sets expectations about how the service will perform.

Service Level Agreement (SLA)

What happens if a published SLO is not met?



Availability

SLI

Once a minute, ssh into the target host, report 1 if it's working 0 if not.

SLO

Uptime 99.9%, i.e. $SLI = 1$ over the last month 99.9% of the time.

SLA

If we don't meet our SLO in one month, you will get exactly one cake.

Latency SLO from the Google Workshop*

SLI

“The proportion of valid requests that were served within $< 1s$ ”

SLO

“90% of valid requests in the past 28 days served in $< 1s$ ”

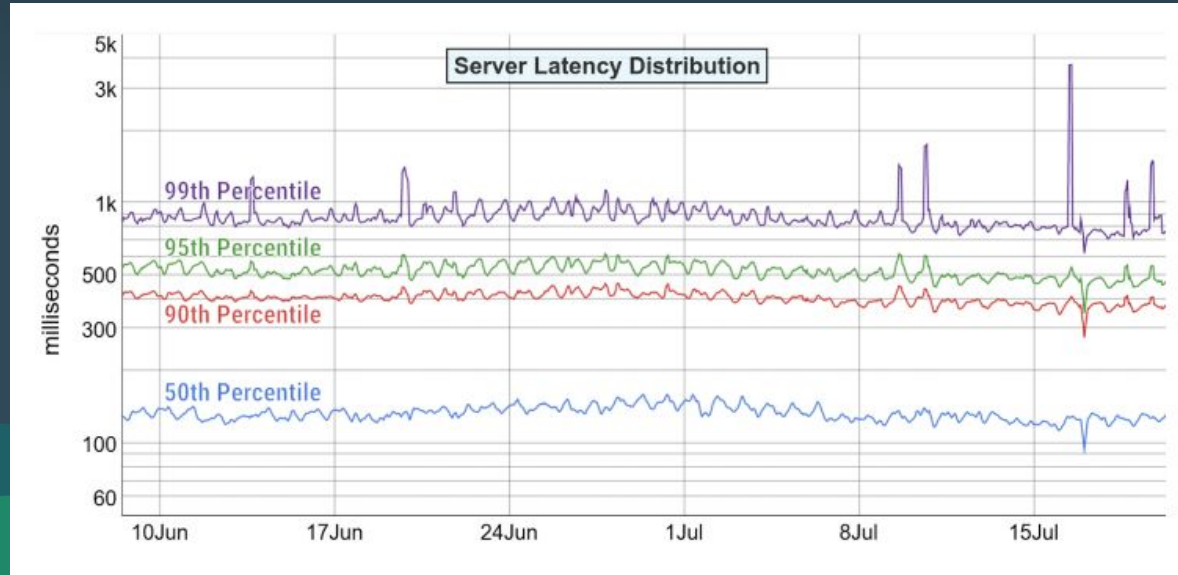
SLA

-

(*) SRECon 2018 : Developing Effective ... Fong-Jones, Bennett, Quinlan, Stockman, Thorne @ Google
[Script](#), p16, we replaced “100ms” by “1s” for the example.

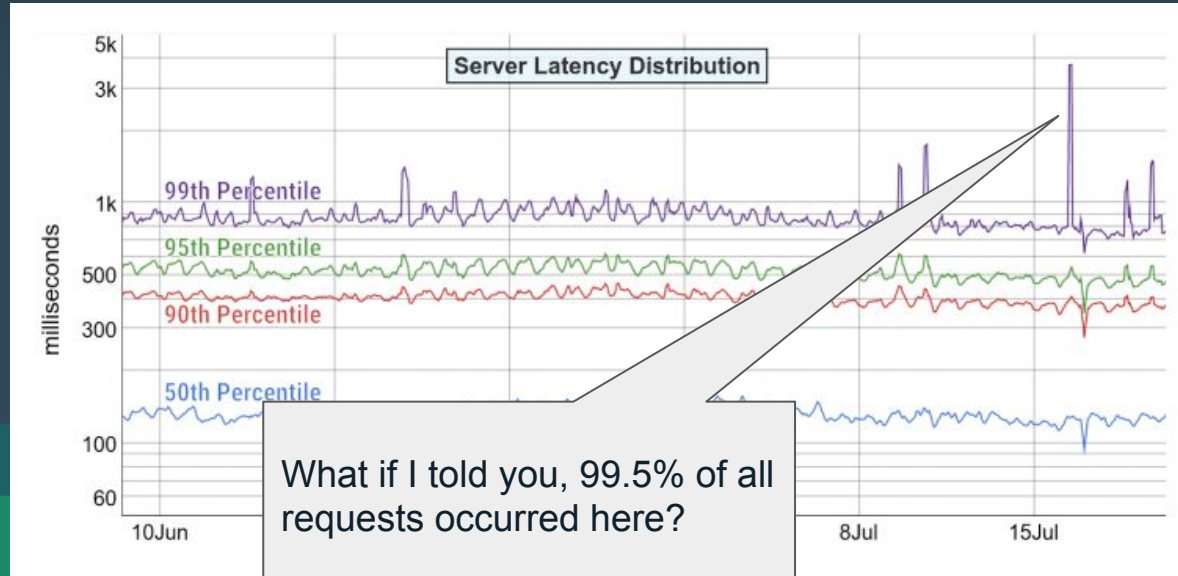
Was the SLO met?

SLO: 90% of valid requests in the past 28 days served in <1s



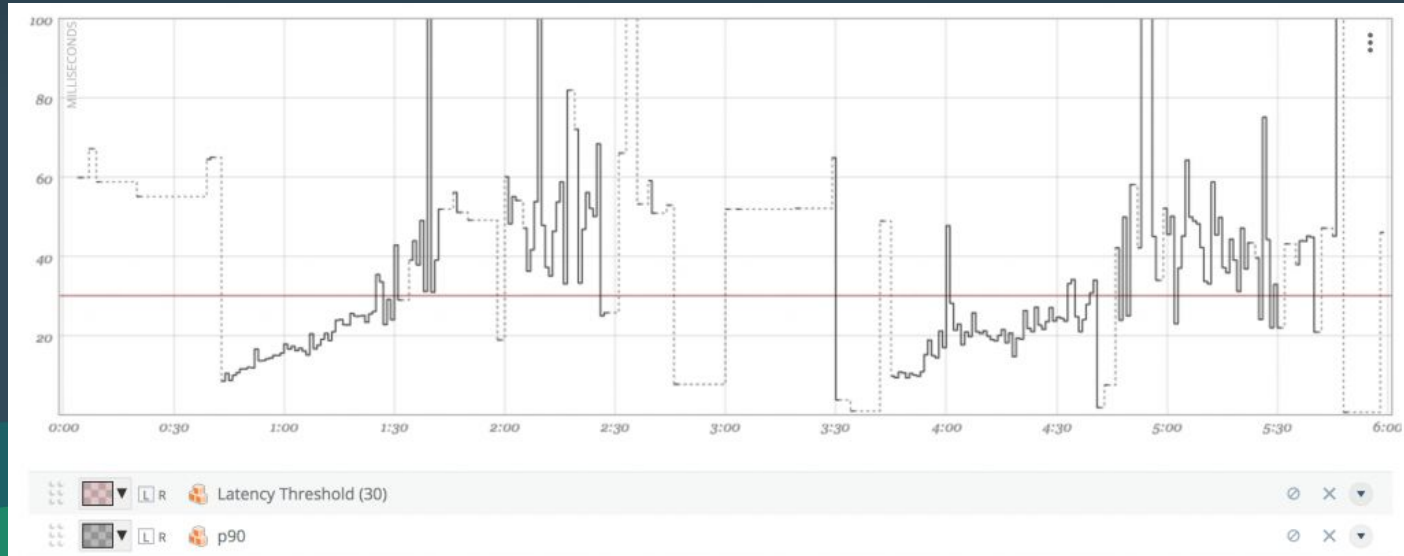
Was the SLO met?

SLO: 90% of valid requests in the past 28 days served in <1s



Was the SLO met?

SLO: 90% of valid requests in the past 6h served in <50ms



Percentile Metrics can't be used for SLOs

For SLOs we need to compute percentiles over ...

(1) multiple weeks of data

(2) multiple nodes (potentially).

But: **Percentiles can't be aggregated.**

Percentiles can't be aggregated

Long story of discussions following my Monitorama 2016 talk:



dan slimmon
@danslimmon

Following

👏 Percentiles 👏 cannot 👏 be 👏 aggregated
– @heinrichhartman #monitorama

1:23 AM - 30 Jun 2016 from Portland, OR



John Rauser @jrauser · 30 Jun 2016

1/ I get annoyed when people say flatly that you can't meaningfully aggregate sample percentiles.

dan slimmon @danslimmon

👏 Percentiles 👏 cannot 👏 be 👏 aggregated 👏 – @heinrichhartman #monitorama

Dear John,

Thank you very much for your comments. I appreciate your passion for this topic. Percentiles are a delicate subtle topic. It's great to have this conversation. I was not able to put my remarks into tweets, so I am using old fashioned "letter" form to reply.

So, I said:

RPubs brought to you by RStudio

You CAN average percentiles!

John Rauser
July 2016

... at least some of the time.

"You can't meaningfully average percentiles."

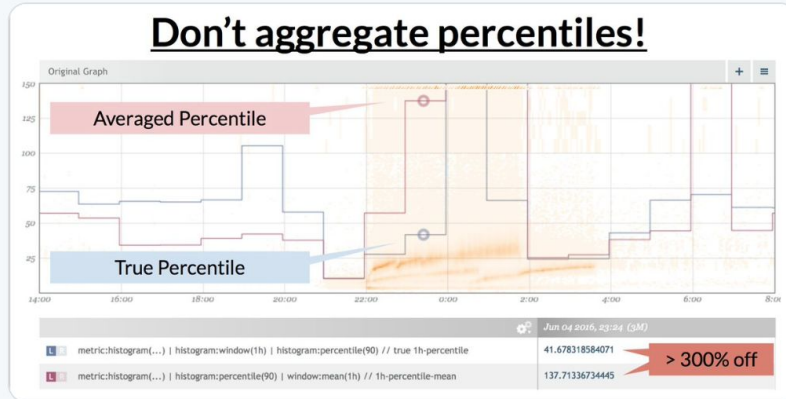
From time to time I hear people say "You can't meaningfully average percentiles." This has always irritated me. Last week I said as much on twitter.

Percentiles can't be aggregated



Heinrich Hartmann 🇪🇺 🇩🇪 @heinrichhartman · 5 Jul 2016

Nice writeup. In practice things look pretty bad, though...



John Rauser

@jrauser

Following

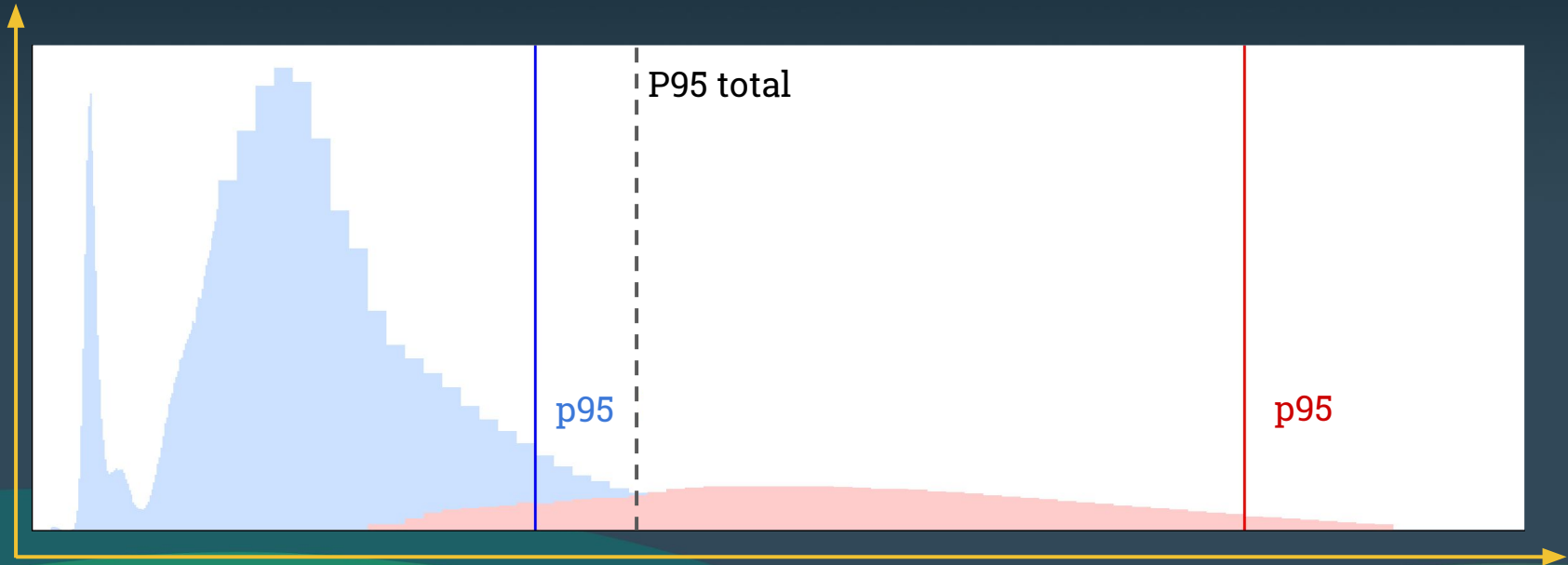
Replying to @heinrichhartman

Yep. It's easy to find real world examples where what seems like a good idea fails.

3:32 AM - 5 Jul 2016

Percentile Aggregation breaks when you need it most

Density



Reddit: https://www.reddit.com/r/devops/comments/941n2k/tsdbs_at_scale_part_one/e3po8d3/

Latency (ms)

@heinrichhartmann

Latency SLOs Done Right

Task

Count all requests over \$period served faster than \$threshold.

Three valid Methods

(1) Log data

(2) Counter Metrics

(3) Histogram Metrics

(1) Latency SLOs via Log Data

```
SELECT count(*) FROM logs  
WHERE  
  time in $period  
AND  
  latency < $threshold
```

(1) Latency SLOs via Log Data

+ Correct, Clean, Easy

- You need to keep all your log data for months

- ... which can get very expensive.

You can do this with: ssh+awk, ELK, Splunk, Honeycomb, etc.

(2) Latency SLOs with Counter Metrics

- Pick a latency \$threshold, e.g. 1s
- Add a metric that counts how many requests were faster than \$threshold
- ... store as e.g. "aws-eu.www22.GET./.lt_1s"
- ... for each node and endpoint you care about
- Sum / Integrate these metrics across nodes, endpoint and time:

```
find("aws-eu.www*.GET.*.lt_1s") | stats:sum() | integrate()
```

(2) Latency SLOs with Counter Metrics



9.5 K / 106 K = 8.9%

Slow Requests

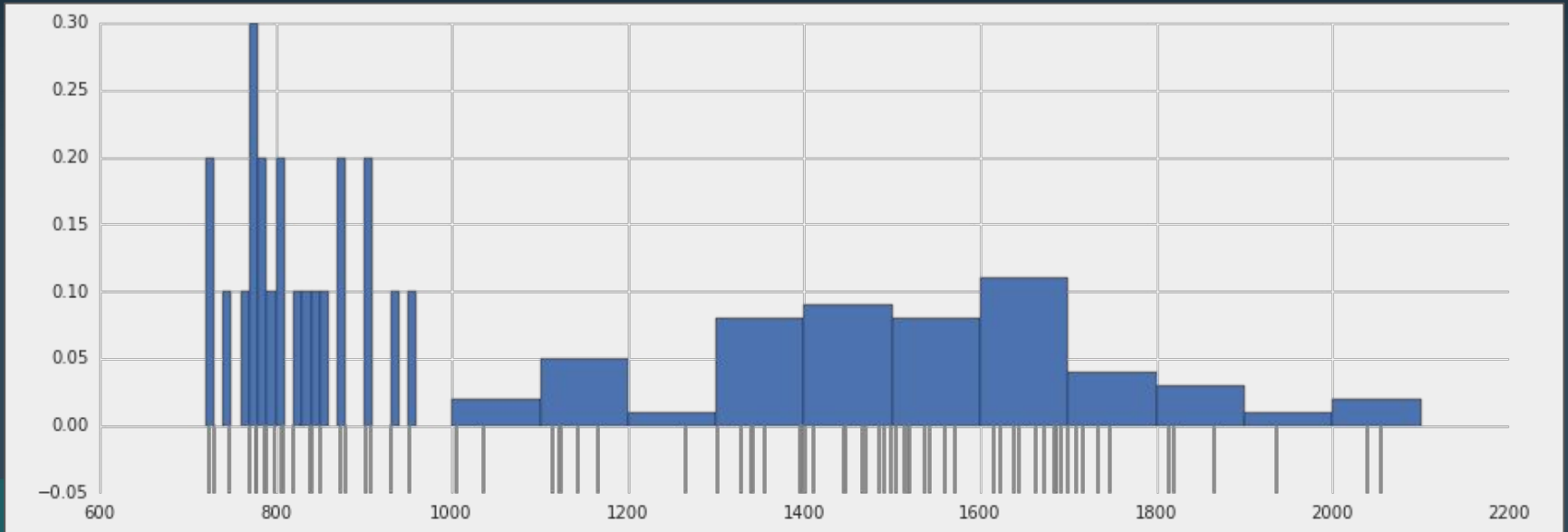
(2) Latency SLOs via Counter Metrics

- + Easy, Correct
- + Cost effective
- + Full flexibility in choosing aggregation intervals
- Need to choose (multiple) latency thresholds upfront

You can do this with:

Prometheus (“Histograms”), Graphite, DataDog, VividCortex

HDR Histograms



HDR Histogram Data Structures

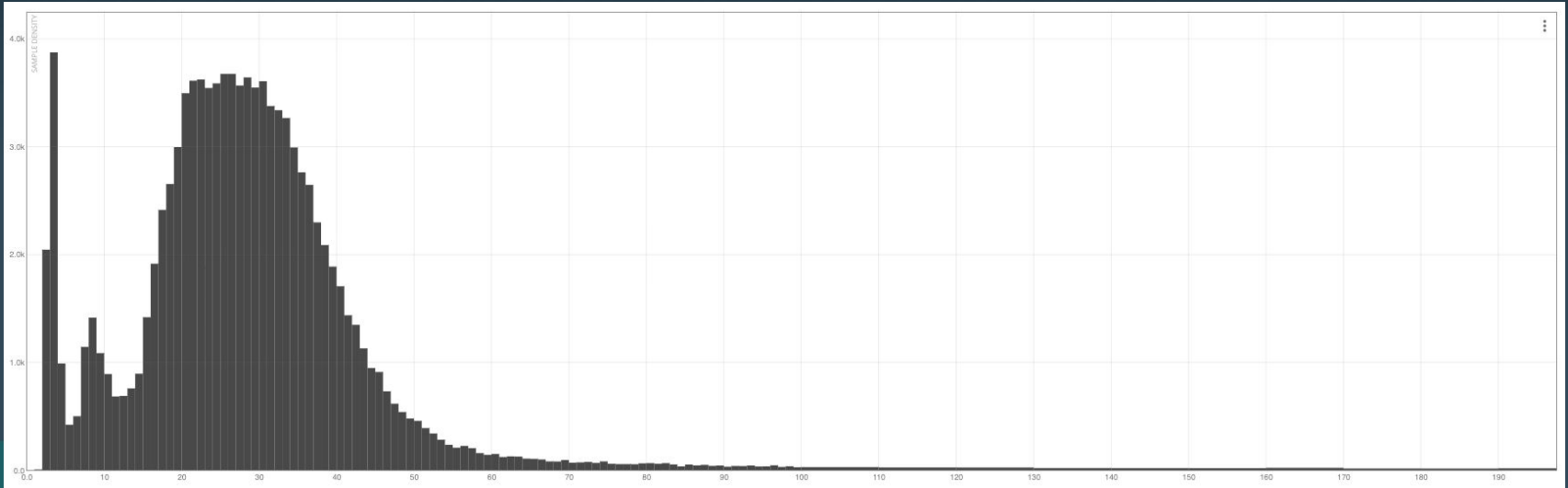
Store Latency distribution in sparse, log-linear histogram data structures:

- Total of 46.081 bins that cover the decimal float range $\pm 10^{\pm 128}$ (HDR)
- One bin for each decimal float number with two significant digits (log-linear):
10,11,12, ... 100,110,120, ..., 1000, 1100, 1200,...
- Only store bins which have entries (sparse encoding)
- Typical size 300b / Histogram even with >100K entries

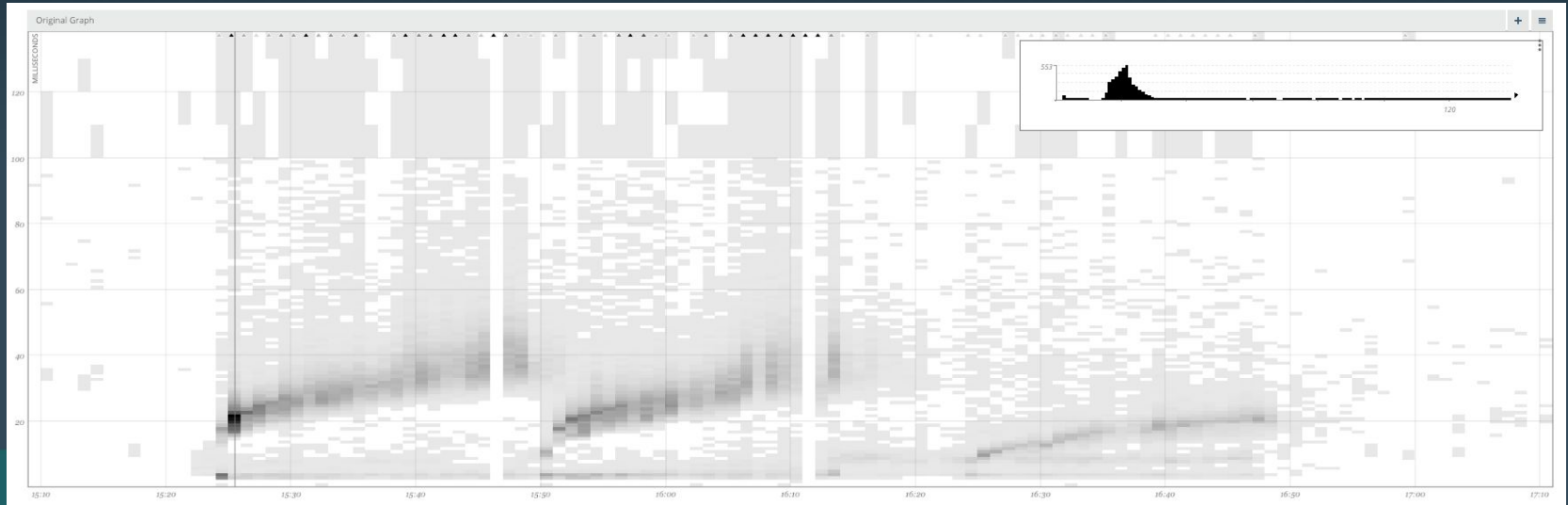
Open Source Implementations:

- hdrhistogram.org (by Gil Tene @ Azul Systems)
- github.com/circonus-labs/libcircllhist (Circonus, IronDB)

HDR Histogram Dataset



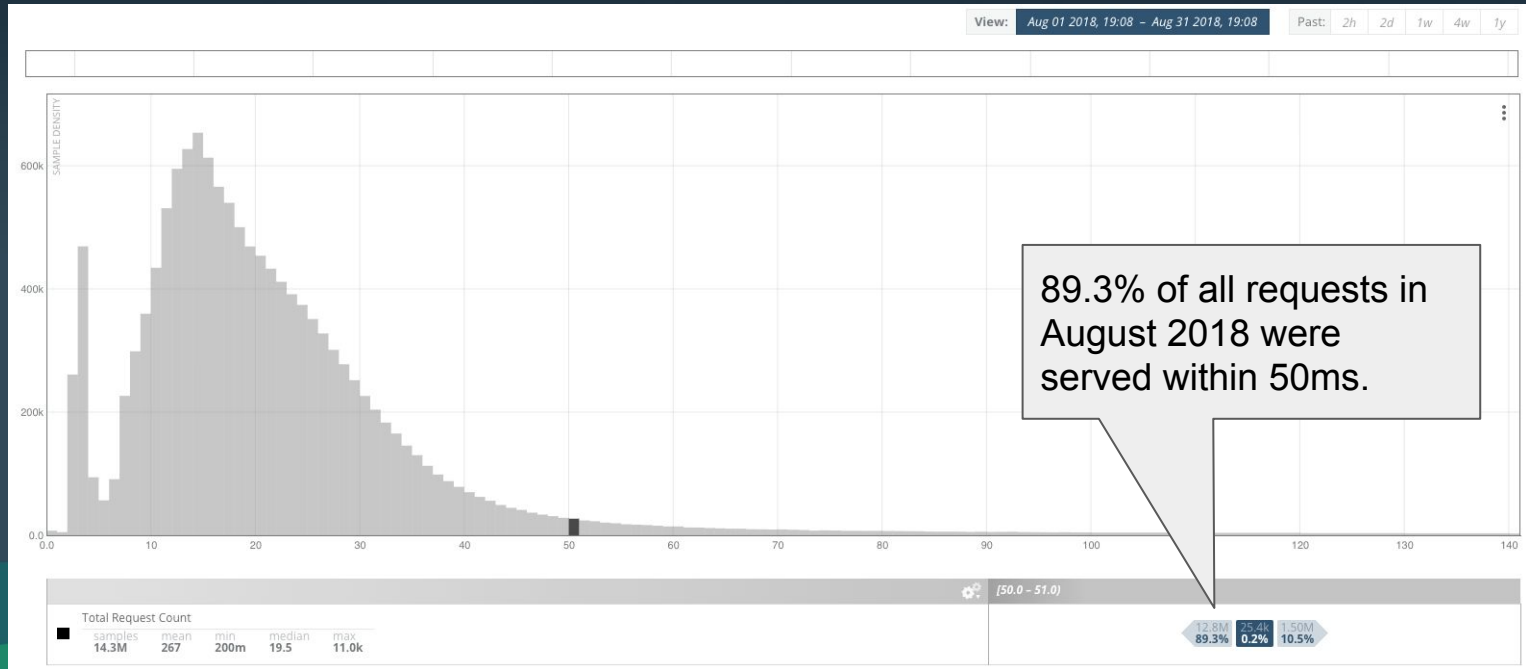
HDR Histogram Metric



(3) Latency SLOs with HDR Histogram Metrics

1. Capture latency information of all relevant APIs as histogram metrics.
2. Aggregate latency histograms over nodes, endpoints and time.
Get **total latency distribution** over SLO timeframe (weeks, months).
3. Count samples in bins below the thresholds to compute SLOs.

(3) Latency SLOs with Histogram Metrics



(3) Latency SLOs with HDR Histograms

- + Full flexibility in choosing thresholds
- + Full flexibility in choosing aggregation intervals and levels
- + Cost effective (300b / Histogram value)
- Need HDR Histogram Instrumentation
- Need HDR Histogram Metric Store

You can do this with: Circonus, IronDB + Graphite / Grafana, Google internal tooling.

Conclusion

Percentile Metrics are not suitable for implementing Latency SLOs.

HDR Histogram Metrics allow you to easily calculate arbitrary Latency SLOs.

If you don't have HDR Histograms, you can do the following:

- Use log data available over the last days to determine sensible latency Thresholds for your service. Which latency is typical / acceptable?
- Add counter metrics for the thresholds.
- Aggregated and Integrate those metrics over weeks and months as needed.