Better Histograms for Prometheus

Björn “Beorn” Rabenstein

Grafana Labs

This is a trilogy:

1. *Secret History of Prometheus Histograms*
   FOSDEM, Brussels, Belgium.
   https://fosdem.org/2020/schedule/event/histograms/
2. *Prometheus Histograms – ~~Past~~, Present, and Future*
   PromCon, Munich, Germany.
   https://promcon.io/2019-munich/talks/prometheus-histograms-past-present-and-future/
3. *Better Histograms For Prometheus*
   KubeCon EU, online, anywhere.
   That's now!

prometheus / tsdb Archived

Unwatch 63 | Star 784 | Fork 165

Code | Issues 0 | Pull requests 14 | Actions | Projects 0 | Wiki | Security | Insights | Settings

Isolation #306

Open **gouthamve** wants to merge 15 commits into `prometheus:master` from `gouthamve:isolation`

Conversation 6 | Commits 15 | Checks 0 | Files changed 7 | +646 −66

gouthamve commented on Mar 19, 2018 • edited by krasi-georgiev Member

A rebase of #105

fixes #260
fixes prometheus/prometheus#1893

Tests are broken and cleanup pending.

This change is Reviewable

brian-brazil added 11 commits on Jun 16, 2017

Reviewers
bwplotka

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone

prometheus / prometheus

Unwatch 1.1k | Unstar 32k | Fork 4.9k

Code | Issues 248 | Pull requests 181 | Actions | Projects | Wiki | Security 1 | Insights | Settings

Releases | Tags

Edit release | Delete

v2.17.0
39e01b3
Verified

Compare

2.17.0 / 2020-03-24

prombot released this on Mar 24

This release implements isolation in TSDB. API queries and recording rules are guaranteed to only see full scrapes and full recording rules. This comes with a certain overhead in resource usage. Depending on the situation, there might be some increase in memory usage, CPU usage, or query latency.

- [FEATURE] TSDB: Support isolation #6841
- [ENHANCEMENT] PromQL: Allow more keywords as metric names #6933
- [ENHANCEMENT] React UI: Add normalization of localhost URLs in targets page #6794
- [ENHANCEMENT] Remote read: Read from remote storage concurrently #6770
- [ENHANCEMENT] Rules: Mark deleted rule series as stale after a reload #6745
- [ENHANCEMENT] Scrape: Log scrape append failures as debug rather than warn #6852
- [ENHANCEMENT] TSDB: Improve query performance for queries that partially hit the head #6676
- [ENHANCEMENT] Consul SD: Expose service health as meta label #5313

https://grafana.com/blog/2020/05/05/how-isolation-improves-queries-in-prometheus-2.17/

How isolation improves queries in Prometheus 2.17

Published: 5 May 2020

Tweet Share

There are instances in life when isolation is actually welcome.

One of those instances pertains to the I in the acronym ACID, which outlines the key properties necessary to maintain the integrity of transactions in a database. The time series database (TSDB) embedded in the Prometheus server has the C (consistency), the D (durability), and – somewhat debatable – the A (atomicity).

But up until and including Prometheus v2.16, it did not have the I (isolation). Lack of isolation in the Prometheus context means that a query running concurrently with a scrape could only see a fraction of the samples ingested in that scrape. It needs the right timing and the right kind of query to make it happen. And even if it does happen, the impact is often hard to notice.

But sometimes, this leads to really weird results. And since the reason is so hard to find and understand, it might hit you really badly if it hits you at all.

Most commonly the problem pertains to histograms. The `histogram_quantile` function goes through the buckets of a histogram, and if some of those buckets are from the current scrape and some from the

by Björn “Beorn” Rabenstein

Recent Posts

How the Cortex and Thanos projects collaborate to make scaling Prometheus better for all

Gardener, SAP's Kubernetes-as-a-service open source project, is moving its logging stack to Loki

Join our sessions on Prometheus and Cortex at PromCon Online

Loki tutorial: How to set up

Mathematically correct aggregation.

By Apdex - Apdex Web site, Fair use, https://en.wikipedia.org/w/index.php?curid=8994240

High frequency sampling feasible.

“What percentage of requests in the last hour got a response in 100ms or less?”

“How many HTTP responses larger than 4kiB were served on 2019-11-03 between 02:30 and 02:45?”

Mathematically correct aggregation. *

By Apdex - Apdex Web site, Fair use, https://en.wikipedia.org/w/index.php?curid=8994240

*

High frequency sampling feasible.

“What percentage of requests in the last hour got a response in 100ms or less?” *

“How many HTTP responses larger than 4kiB were served on 2019-11-03 between 02:30 and 02:45?” *

* If suitable buckets defined.

```
histogram_quantile(0.99, sum(rate(rpc_duration_seconds_bucket[5m])) by (le))
```

```
histogram_quantile(0.99, sum(rate(rpc_duration_seconds_bucket[5m])) by (le))
```

- Accuracy depends on bucket layout.
- Bucketing scheme must be compatible…
  - …across the aggregated metrics.
  - …across the range of the rate calculation.
- ~~Lack of ingestion isolation can wreak havoc.~~

```
httpRequests = prometheus.NewCounterVec(
    prometheus.CounterOpts{
        Name:       "http_requests_total",
        Help:       "HTTP requests partitioned by status code.",
    },
    []string{"status"},
)

httpRequestDurations = prometheus.NewHistogram(prometheus.HistogramOpts{
    Name:    "http_durations_seconds",
    Help:    "HTTP latency distribution.",
    Buckets: []float64{.005, .01, .025, .05, .1, .25, .5, 1, 2.5, 5, 10},
})
```

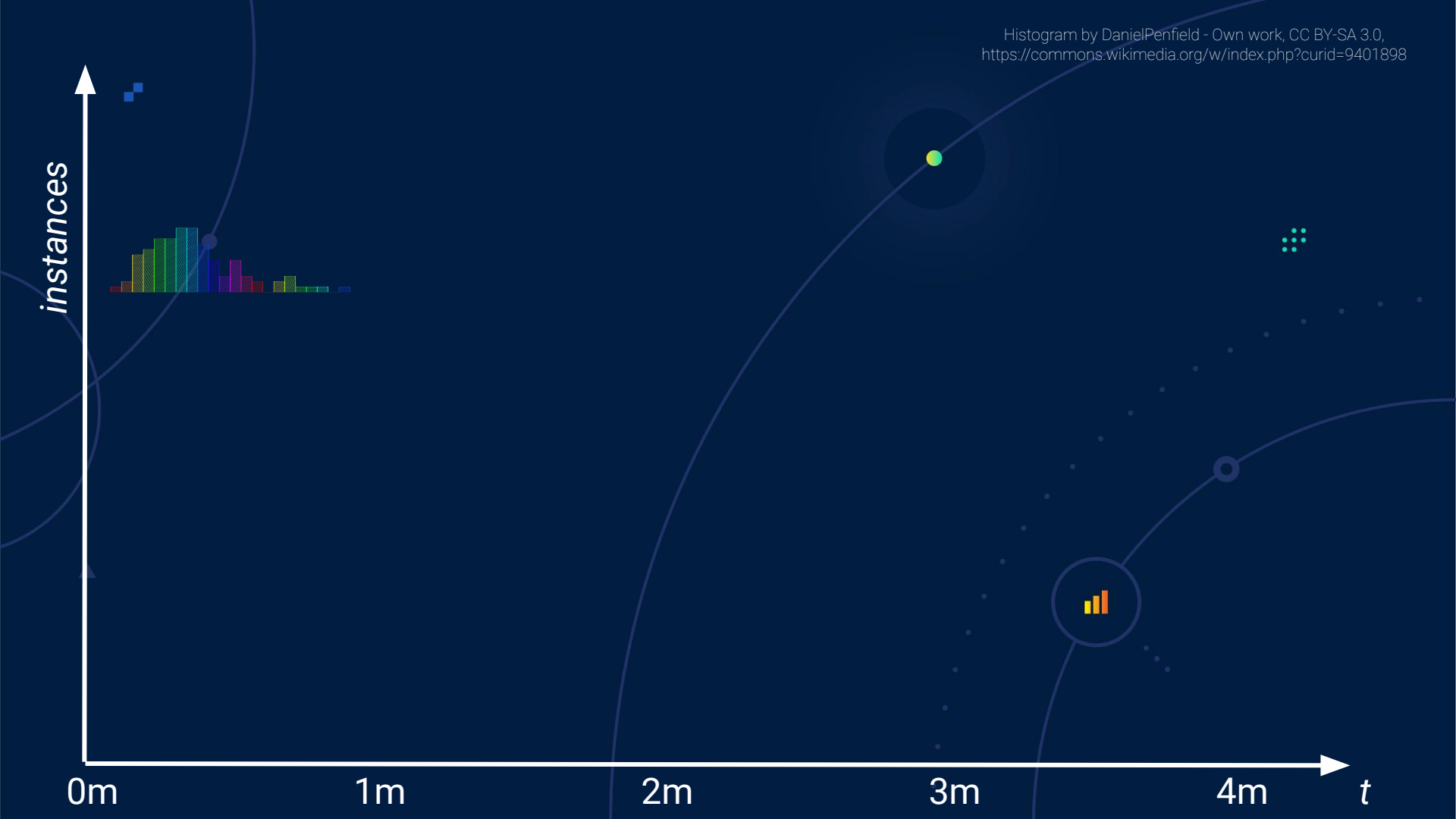
Wishlist

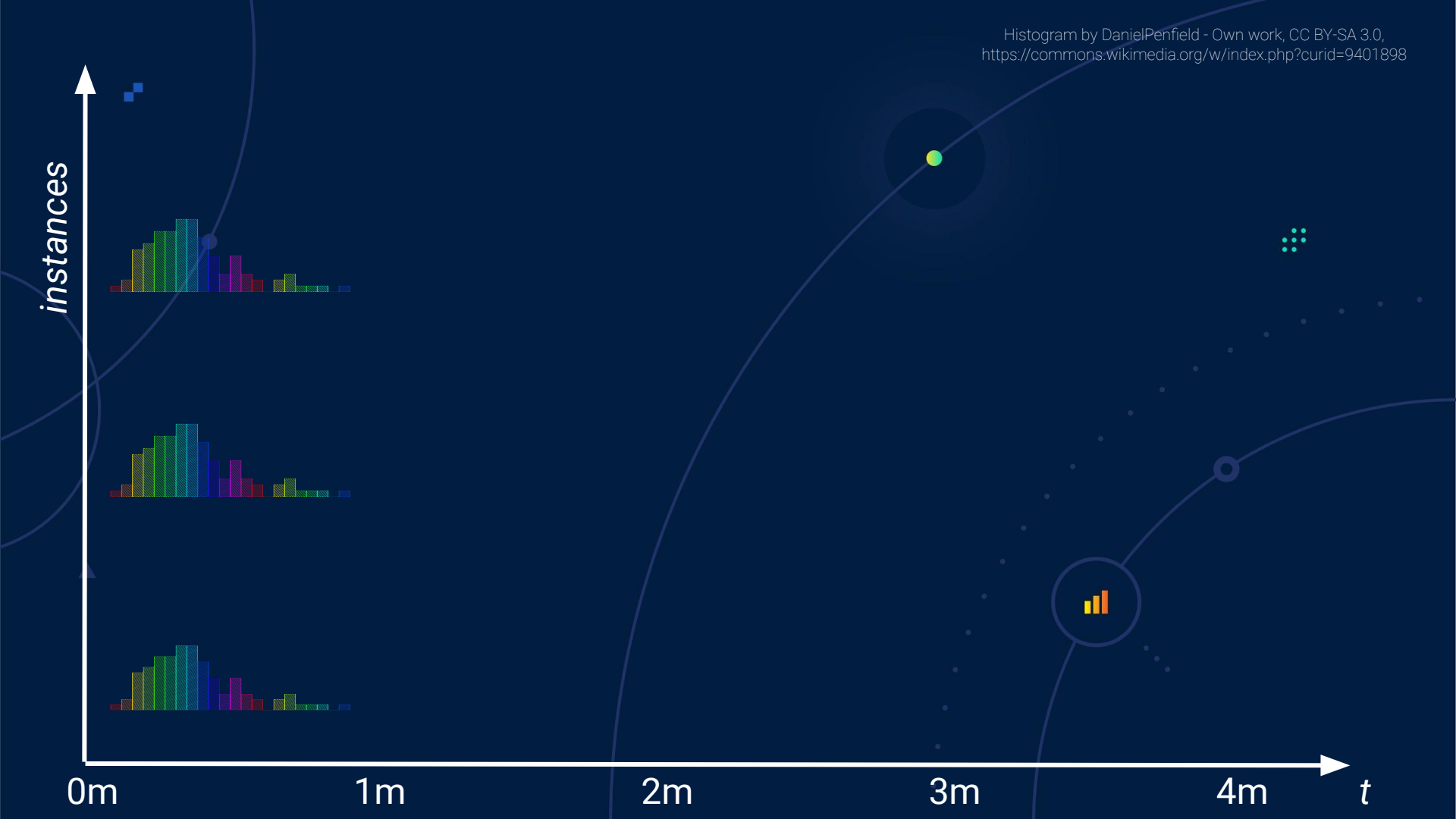
1. Everything that works well now should continue to work well.
2. I never want to configure buckets again.
3. All histograms should always be aggregatable with each other, across time and space.
4. I want accurate quantile and percentage estimations across the whole range of observations.
5. I want all of that at a lower cost than current histograms so that I can finally partition histograms at will.

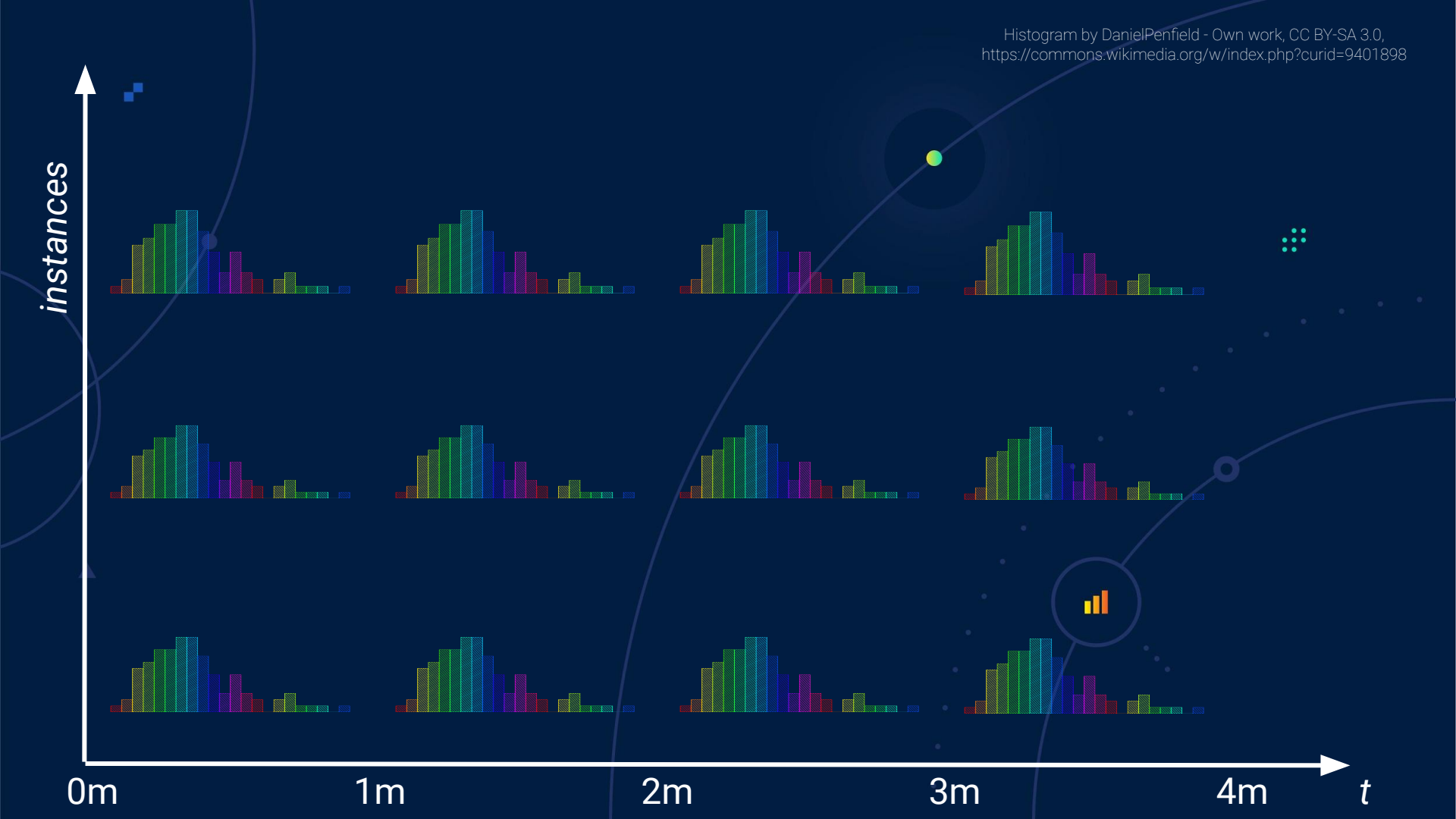
HdrHistogram: http://hdrhistogram.org

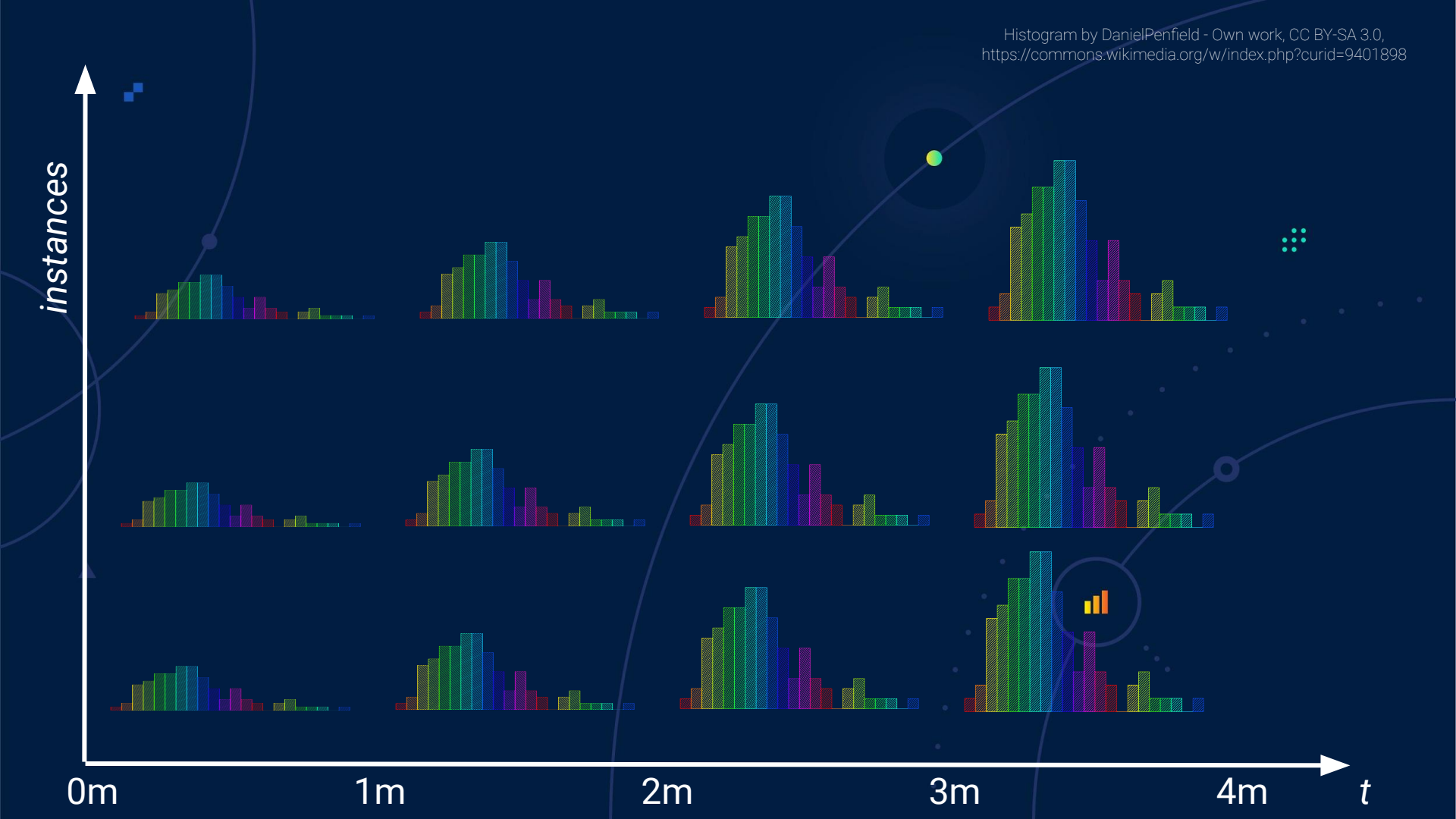
Circonus's CircllhistL https://arxiv.org/abs/2001.06561

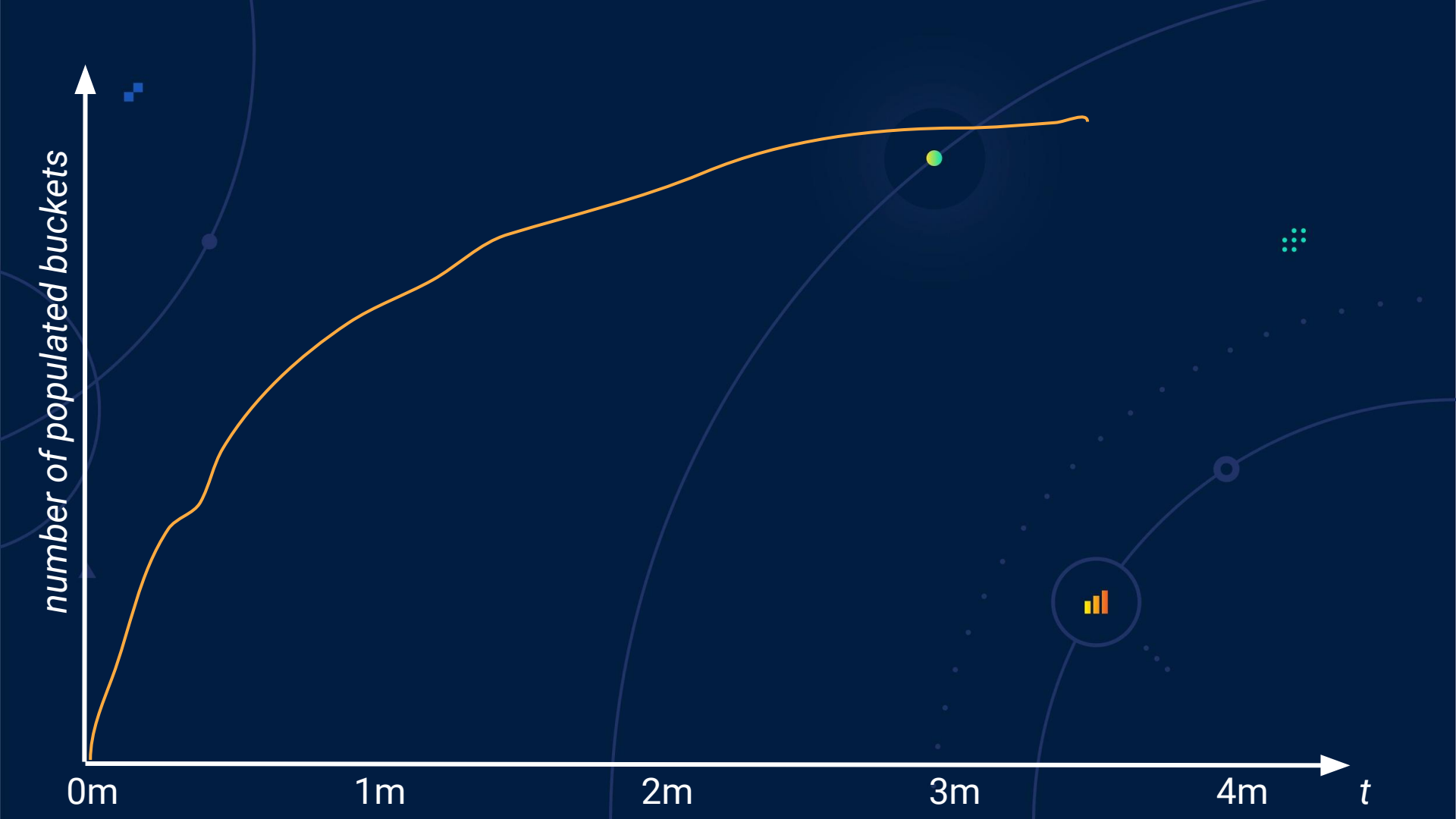
Datadog's DDSketch: https://arxiv.org/abs/1908.10693

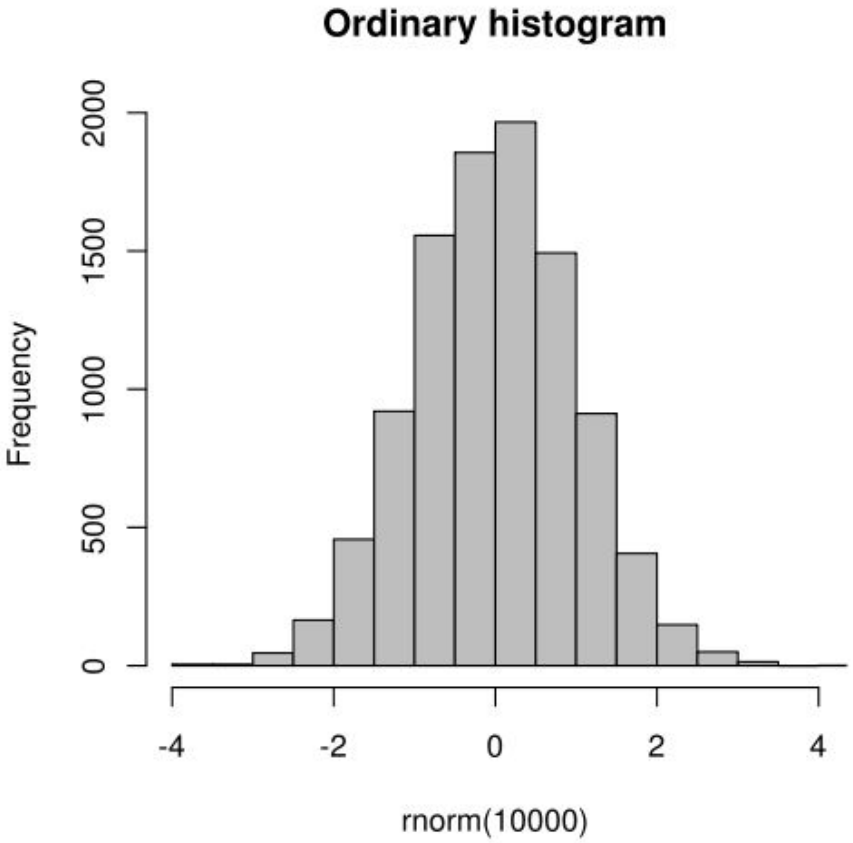


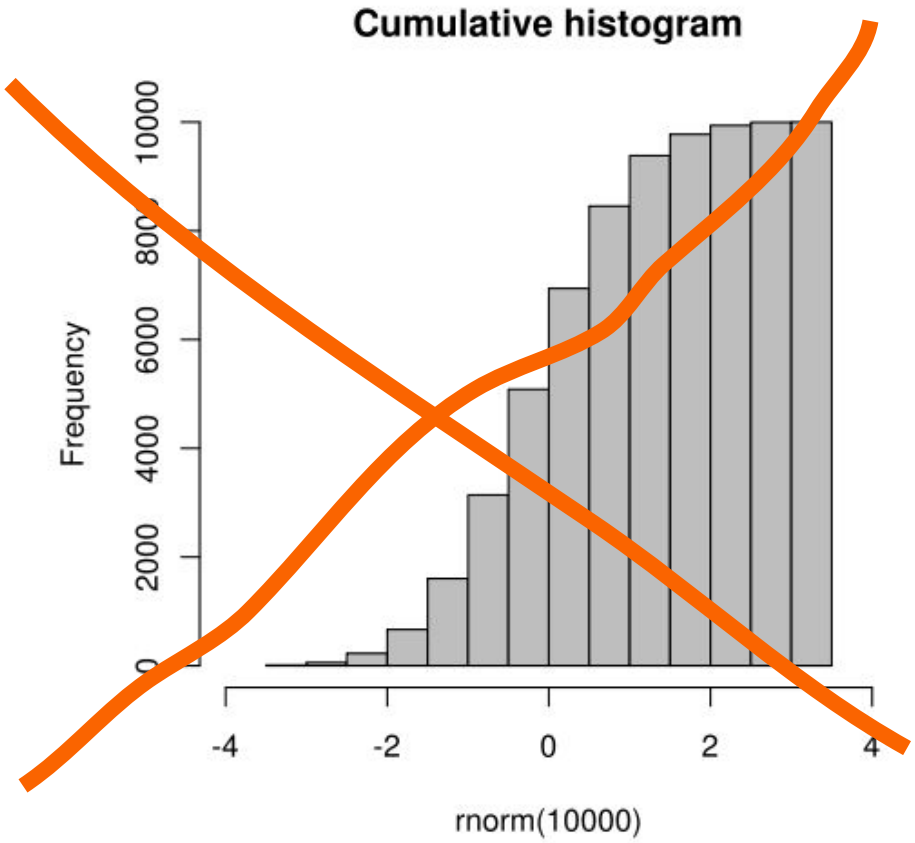


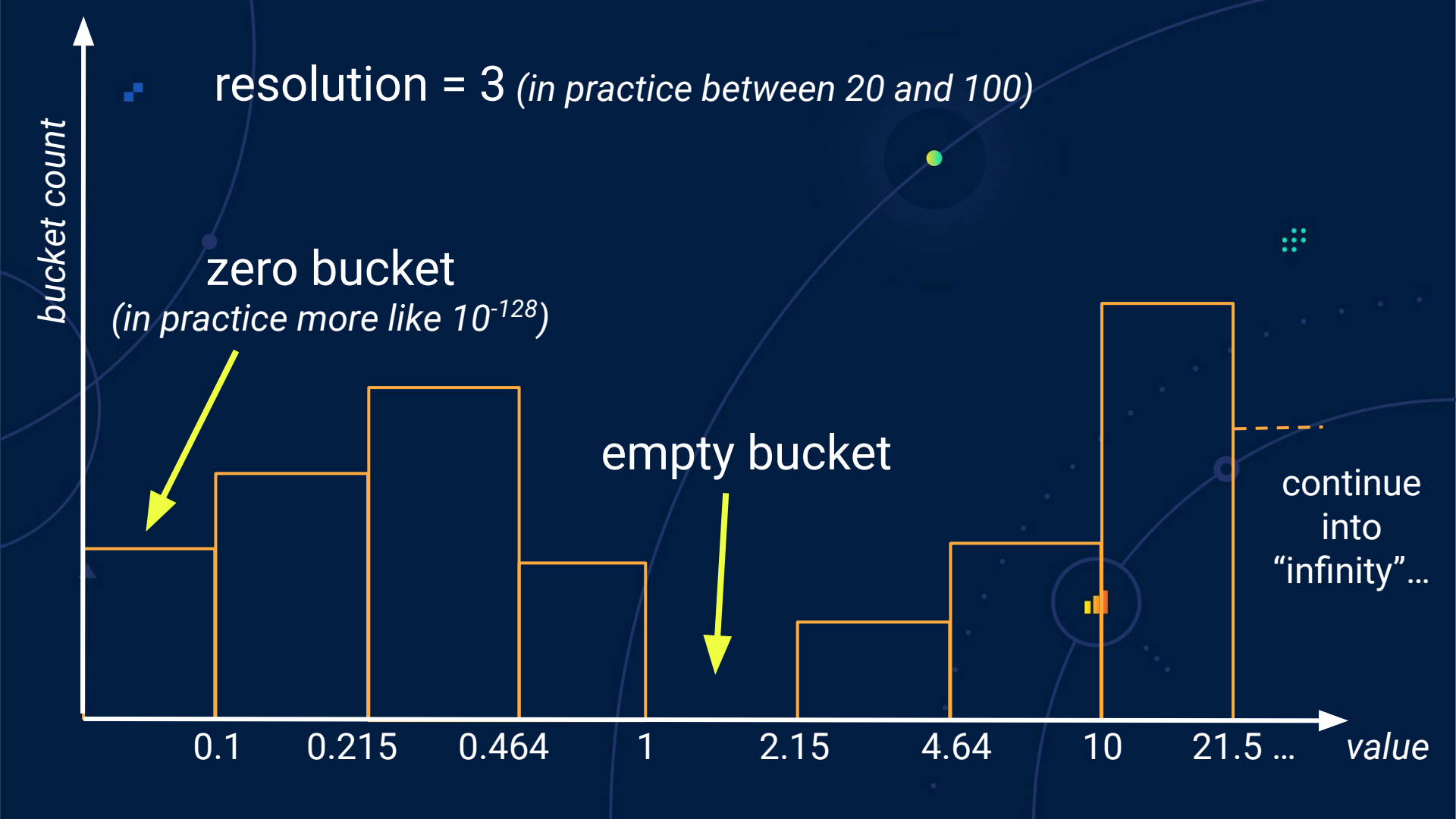

Histogram by DanielPenfield - Own work, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=9401898
instances
0m
1m
2m
3m
4m
t




number of populated buckets
0m
1m
2m
3m
4m
t



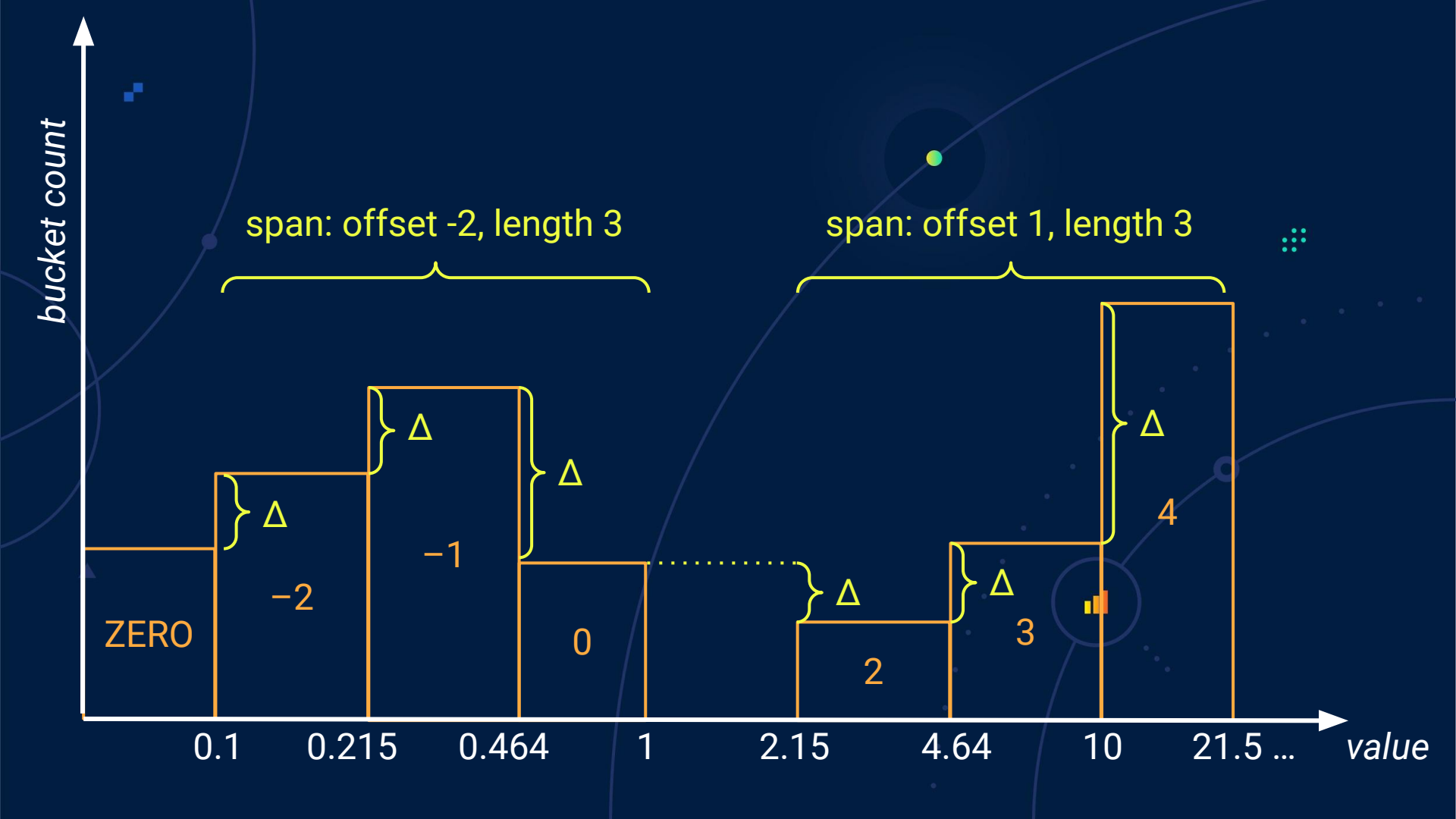



resolution = 3 (in practice between 20 and 100)
bucket count
zero bucket
(in practice more like 10^{-128})
empty bucket
continue into "infinity"...
0.1
0.215
0.464
1
2.15
4.64
10
21.5 ...
value

Real production data

from Grafana Labs' Cortex cluster

Dataset	Observations (duration)	Buckets (res=20)	Buckets (res=100)
Ingester	1,876,573 (2m)	78 (2 spans)	355 (12 spans)
Querier	668,925 (1h)	112 (1 span)	535 (15 spans)


bucket count
span: offset -2, length 3
span: offset 1, length 3
ZERO
−2
−1
0
2
3
4
Δ
0.1
0.215
0.464
1
2.15
4.64
10
21.5 ...
value

Real production data

from Grafana Labs'Cortex cluster

Dataset	Buckets (res=20)	proto size	Buckets (res=100)	proto size
Ingester	78 (2 spans)	244 bytes	355 (12 spans)	898 bytes
Querier	112 (1 span)	317 bytes	535 (15 spans)	1294 bytes

```
# HELP rpc_durations_histogram_seconds RPC latency distributions.
# TYPE rpc_durations_histogram_seconds histogram
rpc_durations_histogram_seconds_bucket{le="-0.00099"} 0
rpc_durations_histogram_seconds_bucket{le="-0.00089"} 0
rpc_durations_histogram_seconds_bucket{le="-0.0007899999999999999"} 0
rpc_durations_histogram_seconds_bucket{le="-0.0006899999999999999"} 2
rpc_durations_histogram_seconds_bucket{le="-0.0005899999999999998"} 13
rpc_durations_histogram_seconds_bucket{le="-0.0004899999999999998"} 43
rpc_durations_histogram_seconds_bucket{le="-0.0003899999999999998"} 186
rpc_durations_histogram_seconds_bucket{le="-0.0002899999999999998"} 554
rpc_durations_histogram_seconds_bucket{le="-0.0001899999999999998"} 1305
rpc_durations_histogram_seconds_bucket{le="-8.999999999999979e-05"} 2437
rpc_durations_histogram_seconds_bucket{le="1.0000000000000216e-05"} 3893
rpc_durations_histogram_seconds_bucket{le="0.00011000000000000022"} 5383
rpc_durations_histogram_seconds_bucket{le="0.00021000000000000023"} 6572
rpc_durations_histogram_seconds_bucket{le="0.00031000000000000002"} 7321
rpc_durations_histogram_seconds_bucket{le="0.00041000000000000002"} 7701
rpc_durations_histogram_seconds_bucket{le="0.00051000000000000003"} 7842
rpc_durations_histogram_seconds_bucket{le="0.00061000000000000003"} 7880
rpc_durations_histogram_seconds_bucket{le="0.00071000000000000003"} 7897
rpc_durations_histogram_seconds_bucket{le="0.00081000000000000004"} 7897
rpc_durations_histogram_seconds_bucket{le="0.00091000000000000004"} 7897
rpc_durations_histogram_seconds_bucket{le="+Inf"} 7897
rpc_durations_histogram_seconds_sum 0.10043870352301096
rpc_durations_histogram_seconds_count 7897
```

plaintext	1676 bytes
protobuf	357 bytes

Storage ideas

- One time series per histogram, not per bucket.
- A sample value is now a sequence of bucket deltas, not a float.
- Double-delta encode them like timestamps, “triple-delta”.
- Adjusted Gorilla-style “varbit” encoding, again like timestamps.

Simulated scrapes

Space needed to store the triple-deltas “varbit”-encoded

Dataset	Scrapes	bytes per histogram (per bucket) [res=20]	bytes per histogram (per bucket) [res=100]
Ingester	80	50 (0.65)	180 (0.51)
Querier	240	70 (0.60)	250 (0.45)

Querying? 🤔

How can we know it will work?

Because others have done similar things already, e.g. FiloDB

https://www.slideshare.net/EvanChan2/histograms-at-scale-monitorama-2019

(slide 23! ❤️)

Wishlist

1. Everything that works well now should continue to work well.
2. I never want to configure buckets again.
3. All histograms should always be aggregatable with each other, across time and space.
4. I want accurate quantile and percentage estimations across the whole range of observations.
5. I want all of that at a lower cost than current histograms so that I can finally partition histograms at will.

Wishlist

1. Everything that works well now should continue to work well.
2. I never want to configure buckets again.
3. All histo… each other, a…
4. I want a… across the who…
5. I want a… ams so that I ca… finally partition histograms at will.

Percentage estimations

SLO: *Respond to 99% of queries in 150ms.*

Bucket boundaries (resolution 100):

147.9ms < x ≤ 151.4ms

Wishlist

1. Everything that works well now should continue to work well.
2. I never want to configure buckets again.

```
his = promauto.NewHistogram(prometheus.HistogramOpts{
    Name:                            "histogram_experiment",
    Help:                            "Test histogram.",
    SparseBucketsResolution:    42,
    SparseBucketsZeroThreshold: 1e-128,
})
```

Wishlist

1. Everything that works well now should continue to work well.
2. I never want to configure buckets again.
3. All histograms should always be aggregatable with each other, across time and space.
4. I want accurate quantile and percentage estimations across the whole range of observations.
5. I want all of that at a lower cost than current histograms so that I can finally partition histograms at will.

Wishlist

1. Everything that works well now should continue to work well.
2. I never want to configure buckets again.
3. All histograms should always be aggregatable with each other, across time and space.
4. I want accurate quantile and percentage estimations across the whole range of observations.
5. I want all of that at a lower cost than current histograms so that I can finally partition histograms at will.

Wishlist

1. Everything that works well now should continue to work well.
2. I never want to configure buckets again.
3. All histograms should always be aggregatable with each other, across time and space.
4. I want accurate quantile and percentage estimations across the whole range of observations.
5. I want all of that at a lower cost than current histograms so that I can finally partition histograms at will.

Very rough results from my experiments and PoC code:
https://github.com/beorn7/histogram_experiments

“Proper” design doc / RFC:
Watch prometheus-developers@googlegroups.com

https://github.com/beorn7/talks

beorn@grafana.com