Observability Data Engineering

A Story About Math, Four Golden Signals, and Business Intelligence

Jack Neely
jjneely@gmail.com

June 19, 2023

DevOps Observability Architect

As a DevOps Observability Architect...

> *What do I monitor?*

Google SRE's ~~Four~~ Five Golden Signals

Traffic	Counter of Units of Work
Errors	Counter of Units of Work with Exceptions
Latency	Timer of the distribution of latencies for each Unit of Work
Saturation	When Pods be scaled up or down
Health	Is the thing up? Does it respond to customers?

As a DevOps Observability Architect...

The ~~Four~~ Five Golden Signals is knowing before the customers do.

> *We need to set alerts for these super special customers.*

Well, if we set our Histograms correctly and record maximum values we will be able to tell when...

> *When a customer calls we need to be able to verify the error they encountered. We'll need a high cardinally solution.*

Umm...those aren't metrics. Where are your traces?

> *Jack, we're an Enterprise!*

Traffic

How to count things

Why Counters Work

> Systems based in cumulative monotonic sums are naturally simpler, in terms of the cost of adding reliability. When collection fails intermittently, gaps in the data are naturally averaged from cumulative measurements.
>
> — OpenTelemetry Data Model Specification

Accurate	Incremented in discrete whole numbers. Never misses an event.
Synchronization	Primitive that allows for multiple observers.
Low Overhead	Easy implementation. No copying or recalling previous values.
Fundamental	Position at time t.

Remembering Physics: First and Second Derivatives

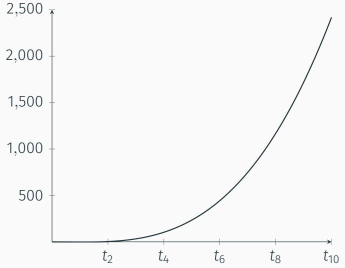


Figure 1: Position: *`requests_total`*

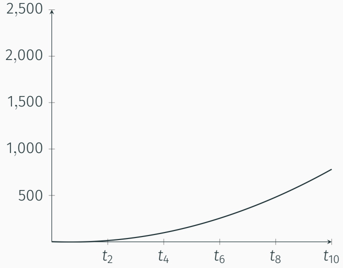


Figure 2: Velocity: *`rate(requests_total[5m])`*

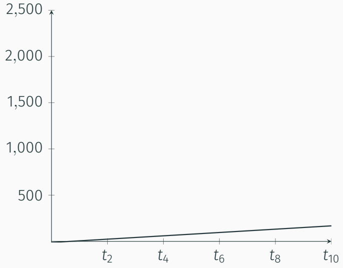
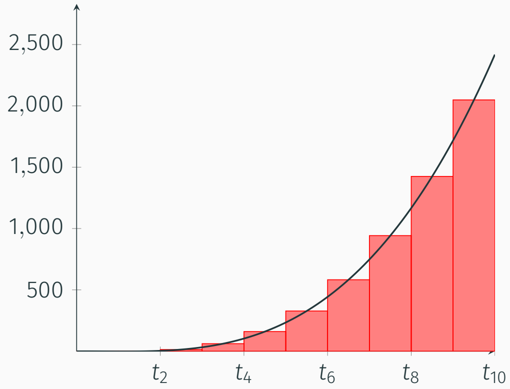


Figure 3: Acceleration: *`deriv(requests:rate5m[5m])`*

Counting Caveats: Riemann Sums



```
interval: 5m
rules:
- record: labels:http_server_requests:rate5m
  expr: >
    sum by (service, namespace, status) (
      rate(http_server_requests_seconds_count{}[5m])
    )
```

Integrate and Build Ratio:

```
1 - (
  sum_over_time(
    sum without (status) (
      labels:http_server_requests:rate5m{
        status=~"5..", service="..."})[7d:5m]
  ) * 300 /
  sum_over_time(
    sum without (status) (
      labels:http_server_requests:rate5m{
        service="..."})[7d:5m]
  ) * 300
)
```

Errors

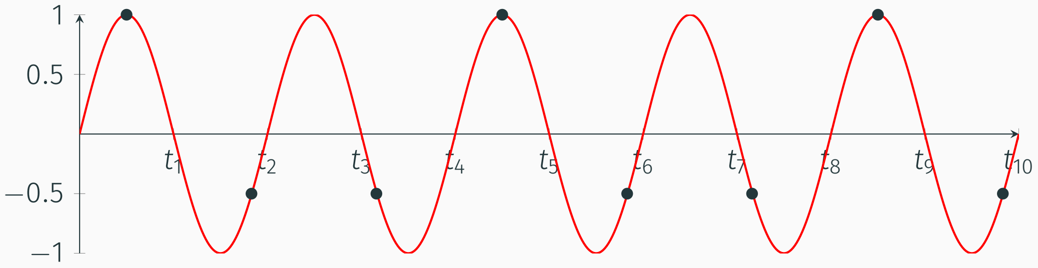
Your CPU Metrics are Wrong and I can Prove It

Measuring CPU Usage Over Time

How do you measure CPU usage of a process?

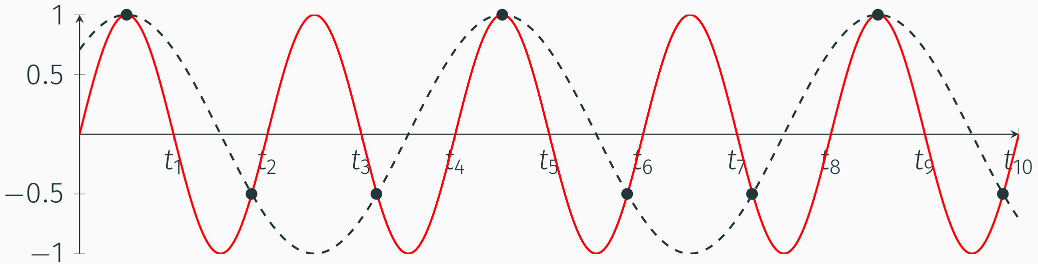
a. Jiffies
b. Percentages
c. Seconds a Process is in the Running State
d. All of the above

Nyquist-Shannon Sampling Theorem



$$ScrapeInterval > 2f$$

Nyquist-Shannon Sampling Theorem: Aliasing



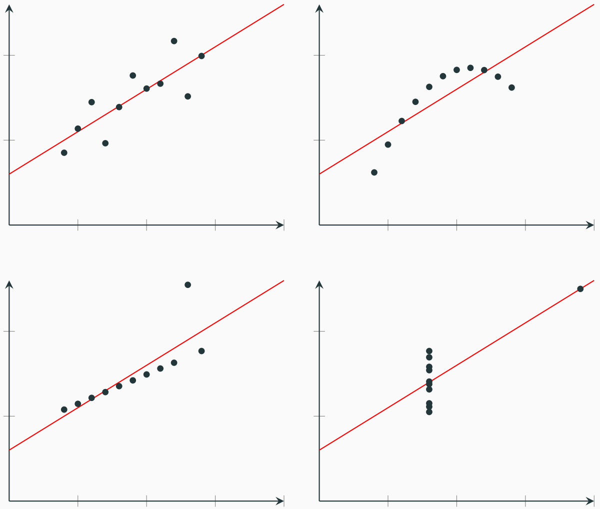
$$ScrapeInterval > 2f$$

Latency

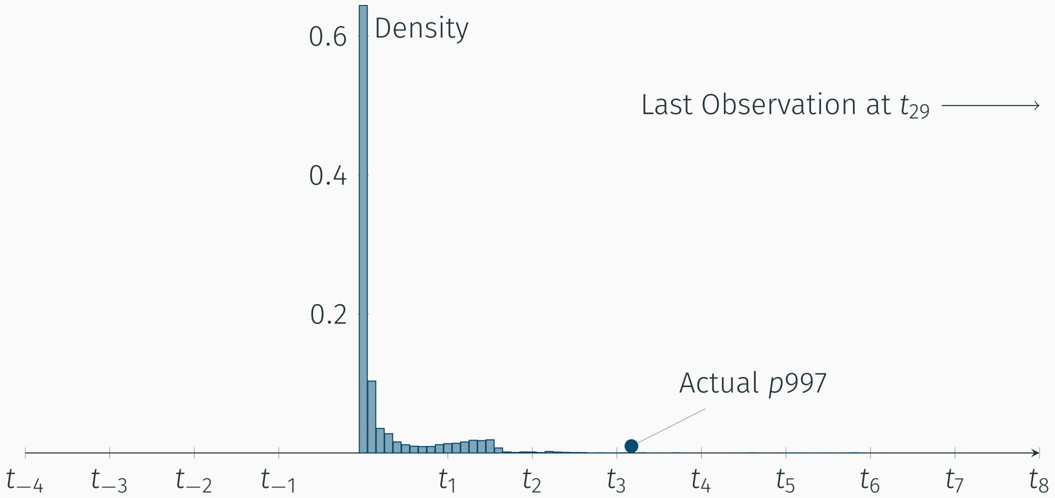
And Other Non-Normal Distributions

Anscomb's Quartet

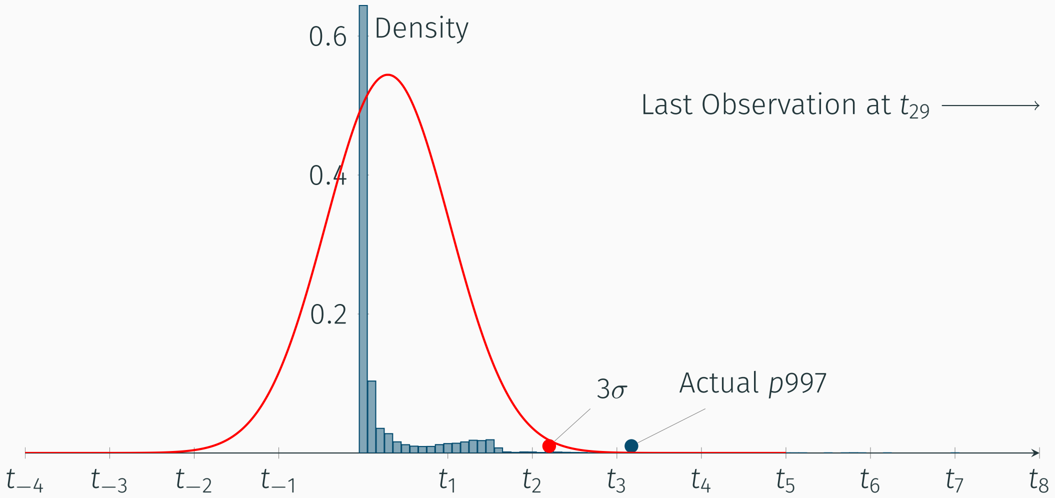
Summary Statistics	
N	11
$\mu\{x_1..x_n\}$	9.0
$\mu\{y_1..y_n\}$	7.5
$\sigma\{x_1..x_n\}$	3.16
$\sigma\{y_1..y_n\}$	1.94
r^2	0.67



Nonstandard Distributions



Nonstandard Distributions



Standard Distribution Curve Formula

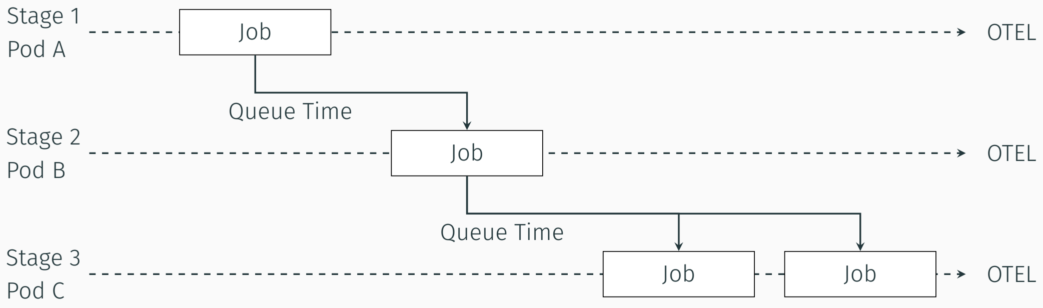
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

σ	Standard Deviation
μ	Mean
e	The base of Natural Logarithm and is about 2.71828
π	Pi! About 3.14159

Saturation

Are We Saturated Yet?

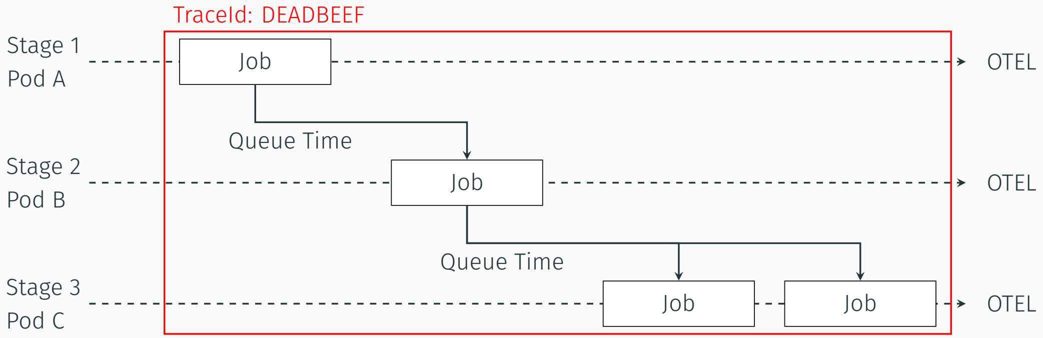
Tracing Pipelines



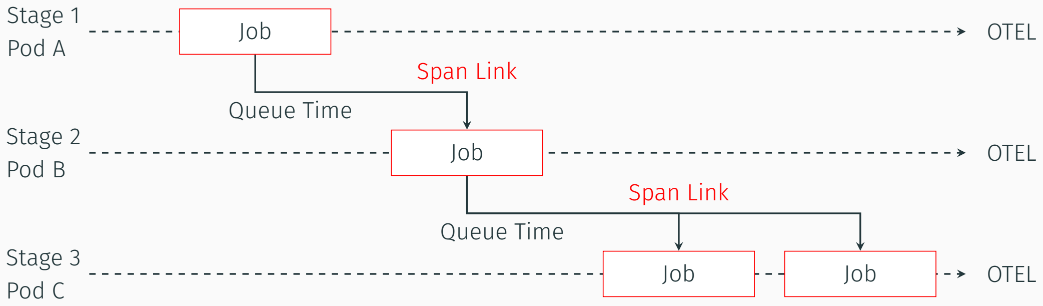
Freshness SLO X% of results are processed in Y time or less over the last Z days.

Saturation SLO X% of results have Y queue time or less over the last Z days.

Tracing Pipelines: How to FAIL



Tracing Pipelines: Using Span Links



Create a TraceId per job and pass context across the bus. Child jobs create a Span Link to reference the TraceId of the parent pipeline job.

Tracing Pipelines: KISS Method

Build a schema and pass meta information along the bus.

```
{
  custId        : int,
  discoveredTs  : Unix Epoch,

  stage1_traceId: string,
  stage1_status : int,
  stage1_startTs: Unix Epoch,
  stage1_stopTs : Unix Epoch,

  stage2_traceId: string,
  stage2_status : int,
  stage2_startTs: Unix Epoch,
  stage2_stopTs : Unix Epoch,

  stage3_traceId: string,
  stage3_status : int,
  stage3_startTs: Unix Epoch,
  stage3_stopTs : Unix Epoch
}
```

Health

Of Your Customers

Mission Impossible

Goal: Per Customer Median and Percentiles

Problem: High Velocity Log/Event Data

Goal: Summarize Per Customer Data Every 15 Minutes

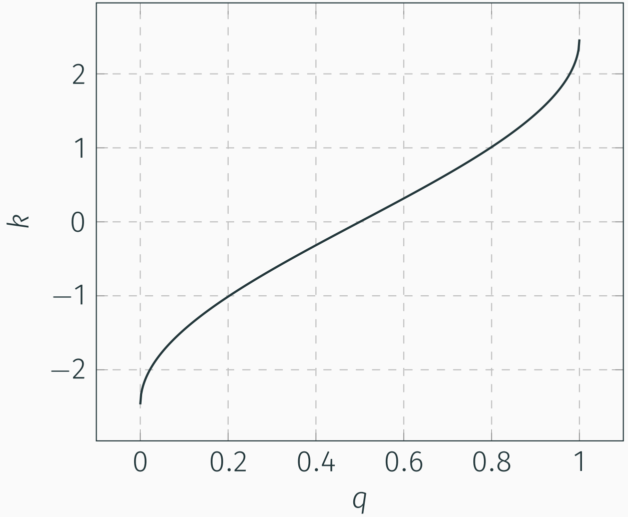
Problem: Calculating 24 Hour (or longer) Percentiles from Rollups

$\{$ *ts*: 2023-06-08T22:15:00, *custId*: 9, *count*: 4, μ: 581, $q(.99)$: 595 $\}$

T-Digest

$$k(q) = \frac{\delta}{2\pi} \sin^{-1}(2q - 1)$$

q	Quantile ($0 - 1$ Inclusive)
k	Scale Factor
δ	Compression Constant
π	Everybody run! It's π again!



Results 24 Hour $q(.99)$ Estimations from 15 Minute Rollups

Overall $q(.99)$ had a 2.32% Error

90% of Per Customer $q(.99)$ had $<$ 50% Error

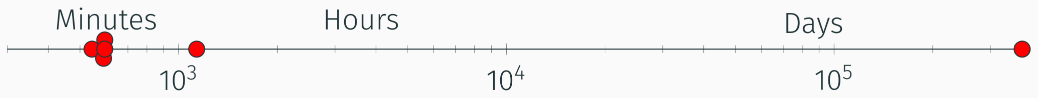


Figure 4: Example of High Error Customer Distribution

Averages Lie

There Are FIVE Golden Signals

Use Quantiles to Understand Latency Spread

Quantiles Cannot Be Combined, But Can Be Estimated

Use the Scientific Method and Mathematically Model Applications

Thank You!

π

Jack Neely jjneely@gmail.com

Podcast: operations.fm