

OBSERVABILITY DATA ENGINEERING

A STORY ABOUT MATH, FOUR GOLDEN SIGNALS, AND BUSINESS INTELLIGENCE

Jack Neely

jjneely@gmail.com

June 20, 2023

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 cardinality solution.*

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

Jack, we're an Enterprise!



TRAFFIC

WHY WE COUNT THINGS

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

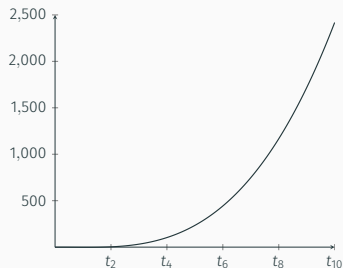


Figure 1: Position:
`requests_total`

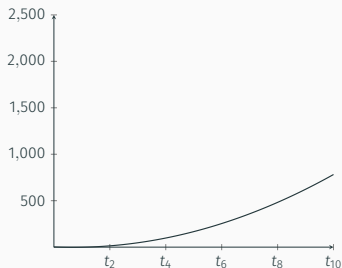


Figure 2: Velocity:
`rate(requests_total[5m])`

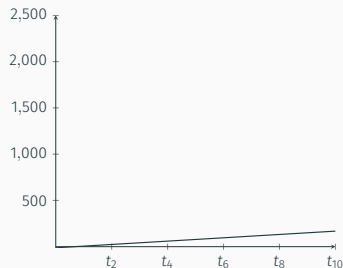
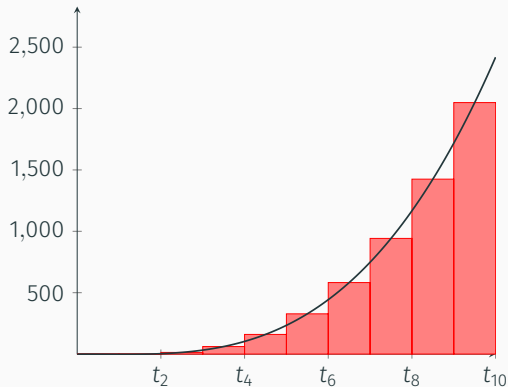


Figure 3: Acceleration:
`deriv(requests:rate5m[5m])`

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
  )
```

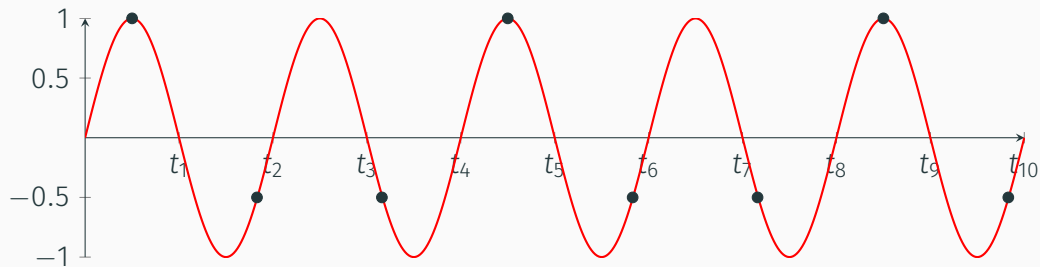

ERRORS

YOUR CPU METRICS ARE WRONG AND I CAN PROVE IT

How do you measure CPU usage of a process?

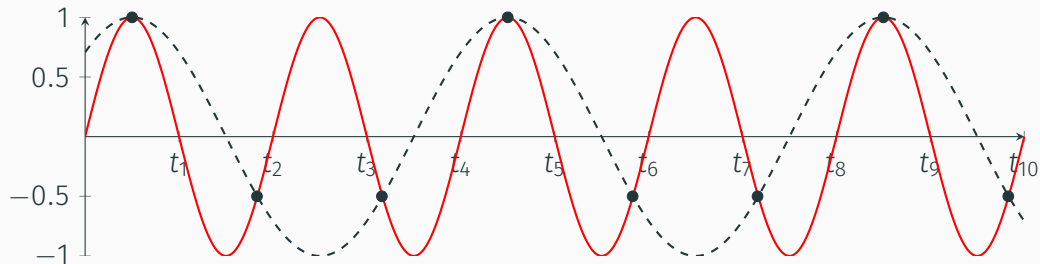
- a. Jiffies
- b. Percentages
- c. Seconds a Process is in the Running State
- d. All of the above

NYQUIST-SHANNON SAMPLING THEOREM



$ScrapeInterval > 2f$

NYQUIST-SHANNON SAMPLING THEOREM: ALIASING



ScrapeInterval > $2f$

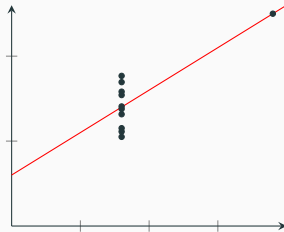
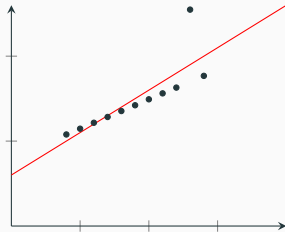
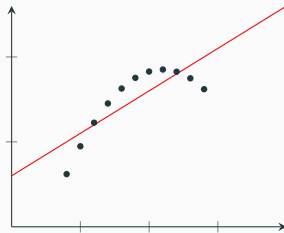
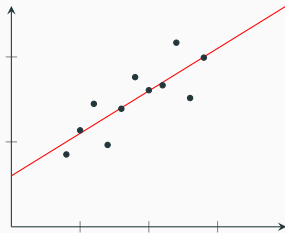
LATENCY

AND OTHER NON-NORMAL DISTRIBUTIONS

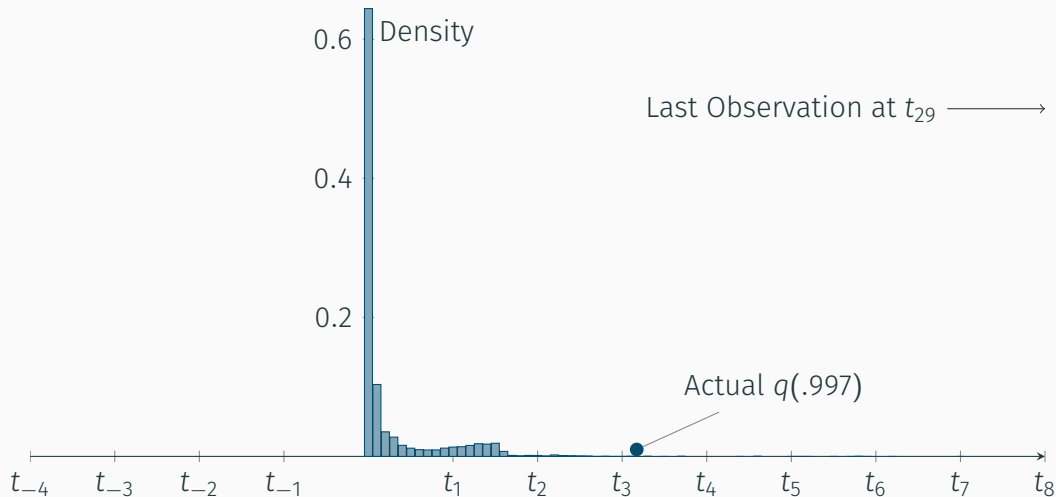
ANSCOMB'S QUARTET

Summary Statistics

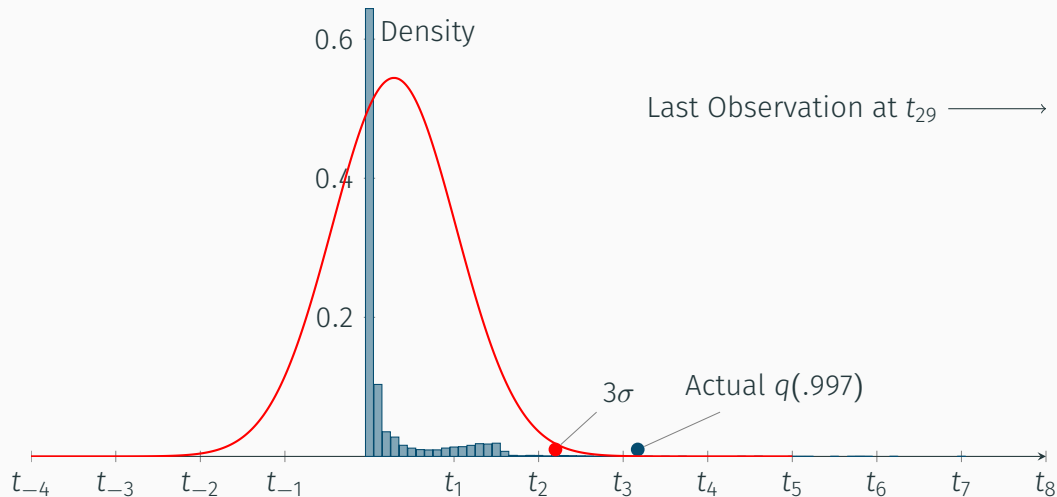
N	11
$\mu\{x_1..x_n\}$	9.0
$\mu\{y_1..y_n\}$	7.5
$\sigma\{x_1..x_n\}$	3.16
$\sigma\{y_1..y_n\}$	1.94
r^2	0.67



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

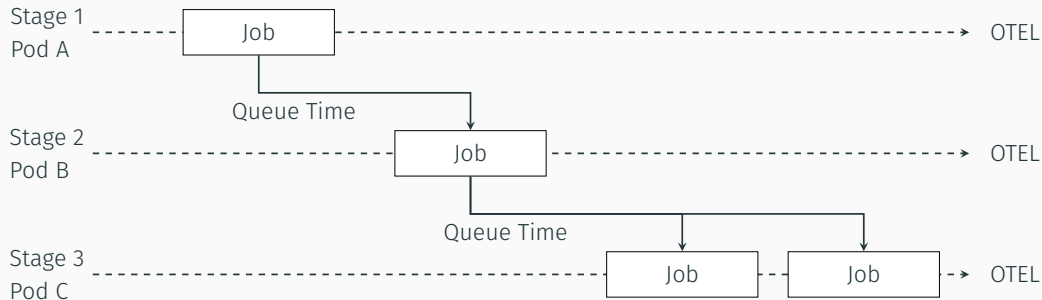
e The base of the Natural Logarithm, about 2.71828

π Pi!

SATURATION

ARE YOU SATURATED YET?

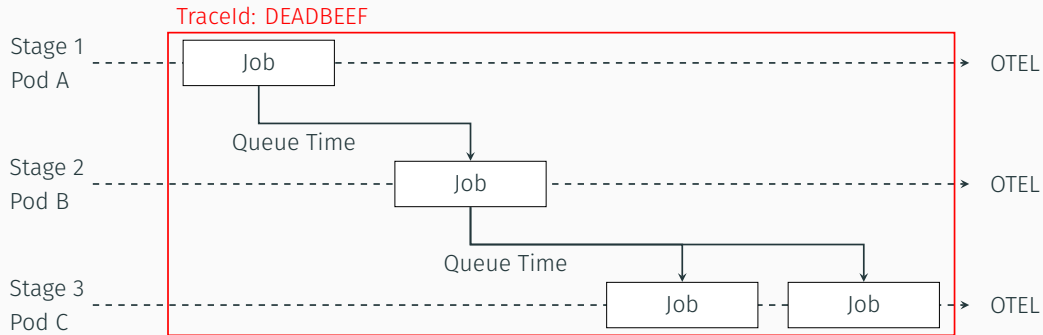
TRACING PIPELINES



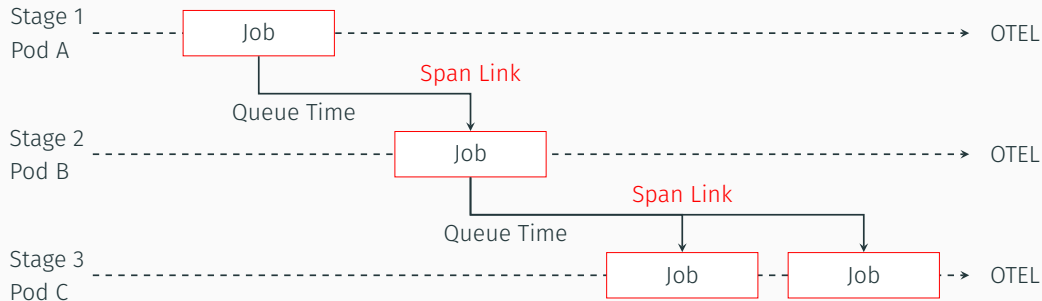
Freshness SLO X% of results are processed in Y time or less over the last Z days.

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

Build a schema and pass meta information along the bus.

Feedback loops for your teams.

```
{  
  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 7 - 30 Day Percentiles from Rollups

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

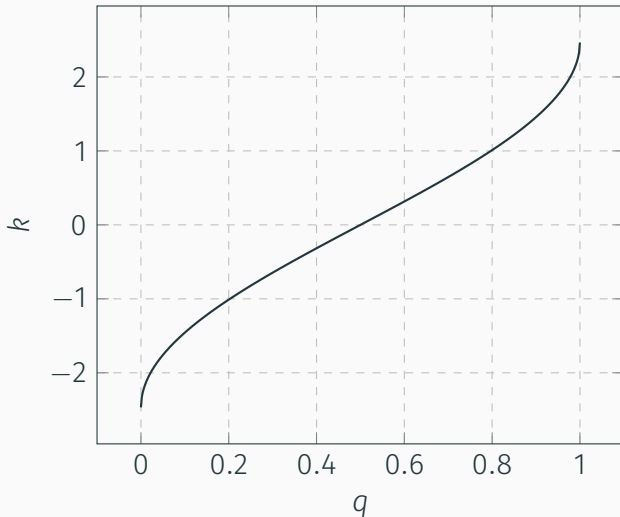
$$k(q) = \frac{\delta}{2\pi} \sin^{-1}(2q - 1)$$

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

Results: 80/20 Rule.

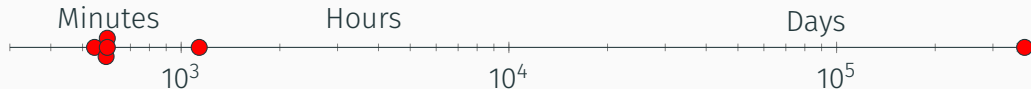


Figure 4: Example of High Error Customer Distribution

Adjusted Hypothesis: Serialized T-Digests as 15 minute rollups will have better accuracy.

Results: 95% of Customer $q(.99)$ Very Accurate

AVERAGES LIE

USE SMART ROLLUPS

THERE ARE FIVE GOLDEN SIGNALS

USE QUANTILES AND MAX TO UNDERSTAND LATENCY SPREAD

USE THE SCIENTIFIC METHOD AND MATHEMATICALLY MODEL APPLICATIONS

THANK YOU!

π

JACK NEELY JJNEELY@GMAIL.COM

PODCAST: OPERATIONS.FM

REFERENCES

Enterprise D Image Credit: PARAMOUNT

T-Digests: <https://github.com/tdunning/t-digest>