Observability Data Engineering

A Story About Math, Four Golden Signals, and Business Intelligence
or
Finding π in Observability

Jack Neely
jjneely@gmail.com

October 16, 2023

DevOps Observability Architect

Goals

Anti-Goals

- Define “Observability”
- Argue “Monitoring” vs “Observability”
- Sell You Things

Actual Goals

- Show the Rabbit Hole
- Be Excited about Math and Engineering
- Share Techniques I Use Everyday
- Observability in the Enterprise

Good Observability is the gateway drug to Data Science. Artificial Intelligence is just Data Science on steroids.

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

Why we 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

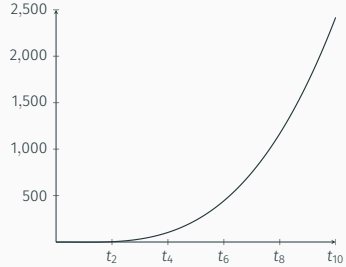


Figure 1: Position: `requests_total`

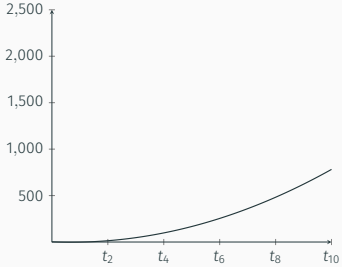


Figure 2: Velocity: `rate(requests_total[5m])`

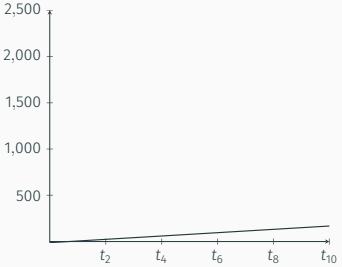
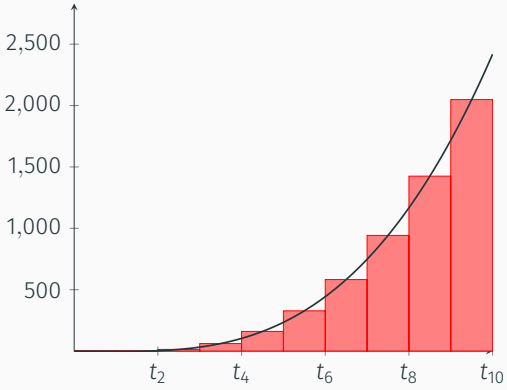


Figure 3: Acceleration: `deriv(requests:rate5m[5m])`

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

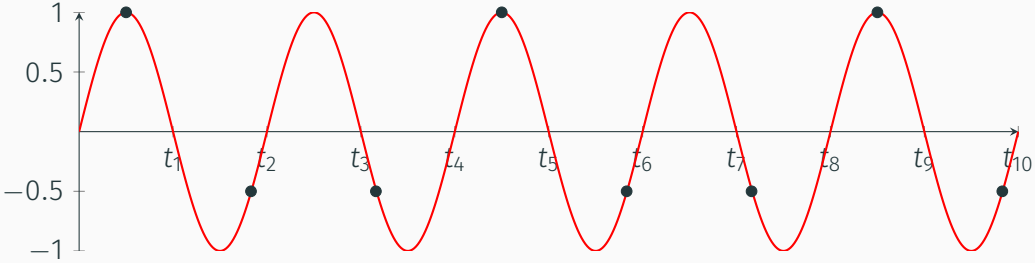
Your CPU Metrics are Wrong and I can Prove It

Measuring CPU Usage Over Time

How do you measure CPU usage of a process?

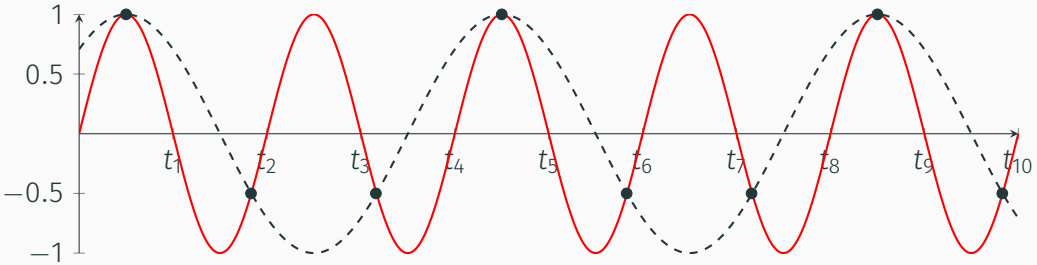
a. Jiffies
b. Percentages
c. Seconds a Process is in the Running State
d. All of the above

Nyquist-Shannon Sampling Theorem



$ScrapeInterval > 2f$

Nyquist-Shannon Sampling Theorem: Aliasing



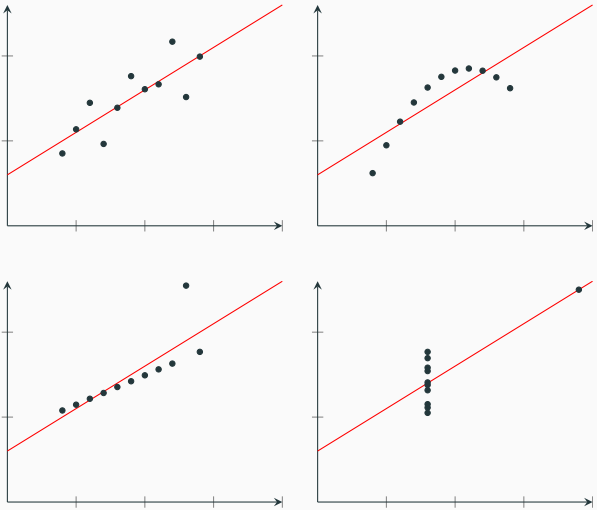
$ScrapeInterval > 2f$

Latency

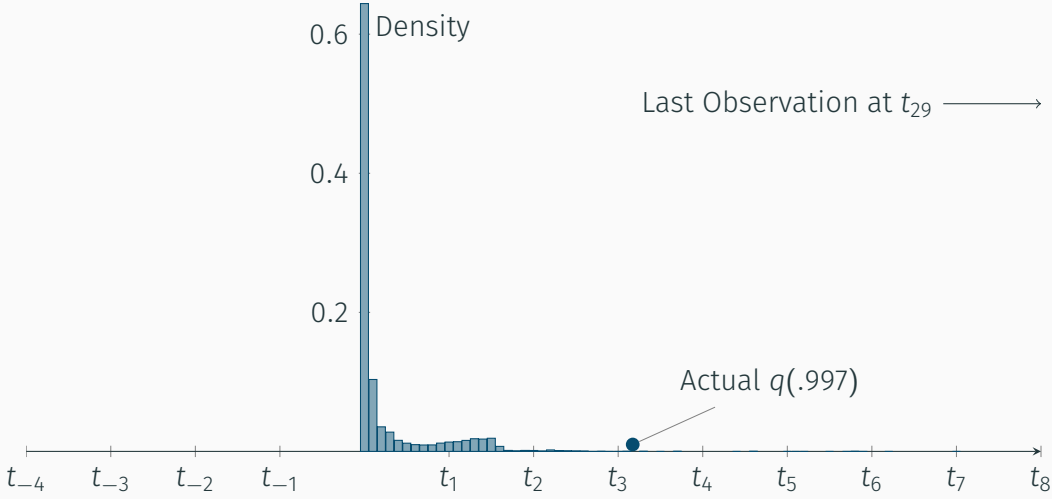
And Other Non-Normal Distributions

Anscomb's Quartet

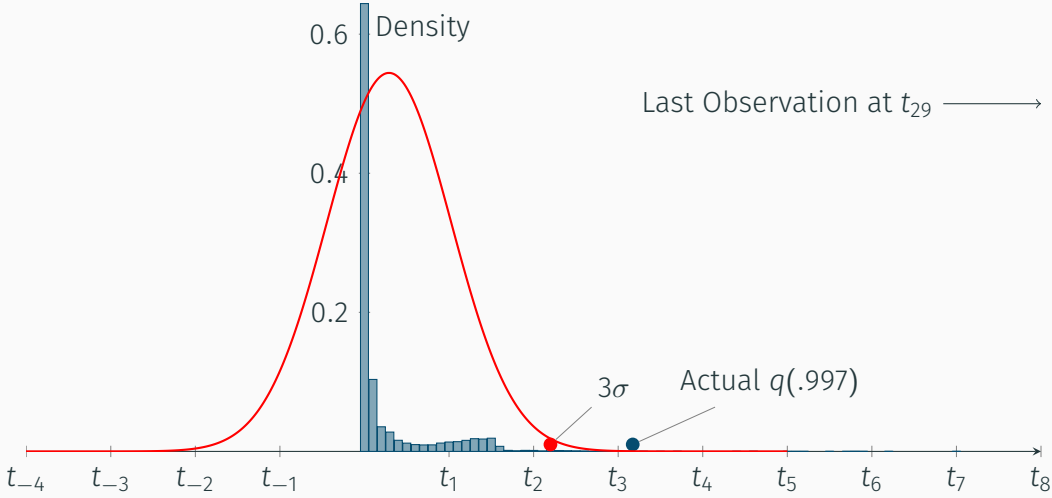
Summary Statistics	
N	11
$\mu\{x_1..x_n\}$	9.0
$\mu\{y_1..y_n\}$	7.5
$\sigma\{x_1..x_n\}$	3.16
$\sigma\{y_1..y_n\}$	1.94
r^2	0.67



Nonstandard Distributions



Nonstandard Distributions



Standard Distribution Curve Formula

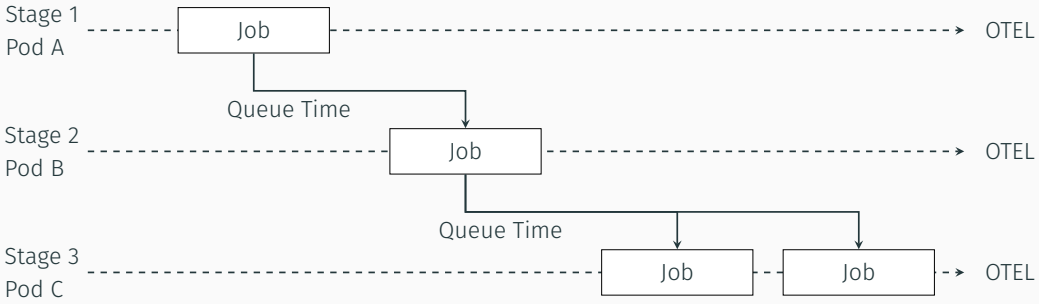
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- σ Standard Deviation
- μ Mean
- e The base of the Natural Logarithm, about 2.71828
- π Pi!

Saturation

Are You Saturated Yet?

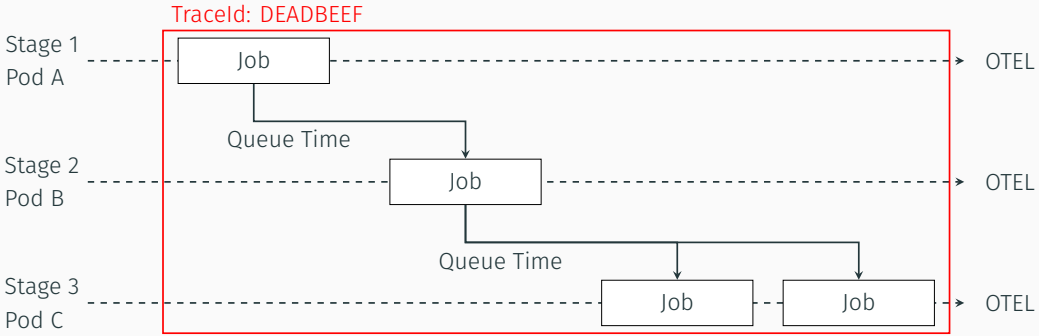
Tracing Pipelines



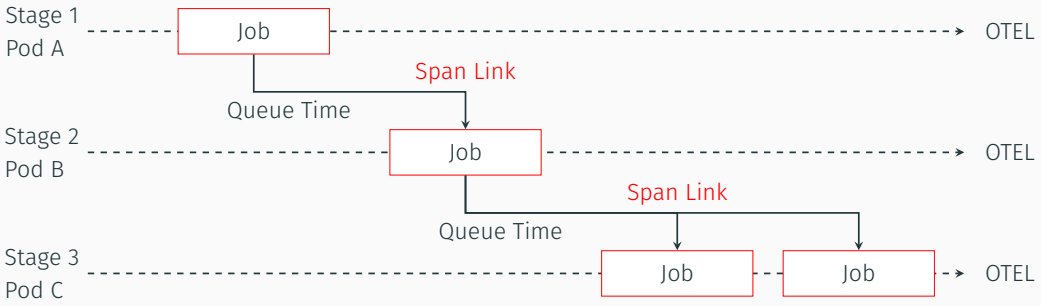
Freshness SLO X% of results are processed in Y time or less over the last Z days.

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 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.

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 and 30 Day Percentiles from Summaries

Requirements for both **Robust** and **Aggregatable** data.

Raw Pipeline Events per Customer


```
{"@timestamp": "2023-06-08T22:15:01.52Z", "cId": 10413, "startTs": 1686262500079, "duration": 564}
{"@timestamp": "2023-06-08T22:15:01.52Z", "cId": 10413, "startTs": 1686262500079, "duration": 764}
{"@timestamp": "2023-06-08T22:15:02.39Z", "cId": 40529, "startTs": 1686262500964, "duration": 522}
{"@timestamp": "2023-06-08T22:15:02.39Z", "cId": 40529, "startTs": 1686262500964, "duration": 712}
{"@timestamp": "2023-06-08T22:15:02.63Z", "cId": 10275, "startTs": 1686262501555, "duration": 652}
{"@timestamp": "2023-06-08T22:15:06.31Z", "cId": 40379, "startTs": 1686262505979, "duration": 157}
{"@timestamp": "2023-06-08T22:15:06.31Z", "cId": 40379, "startTs": 1686262505979, "duration": 242}
{"@timestamp": "2023-06-08T22:15:06.44Z", "cId": 70033, "startTs": 1686262505370, "duration": 539}
{"@timestamp": "2023-06-08T22:15:06.44Z", "cId": 70033, "startTs": 1686262505370, "duration": 539}
{"@timestamp": "2023-06-08T22:15:06.44Z", "cId": 70033, "startTs": 1686262505370, "duration": 539}
{"@timestamp": "2023-06-08T22:15:06.44Z", "cId": 70033, "startTs": 1686262505370, "duration": 629}
{"@timestamp": "2023-06-08T22:15:06.44Z", "cId": 70033, "startTs": 1686262505370, "duration": 637}
{"@timestamp": "2023-06-08T22:15:06.44Z", "cId": 70033, "startTs": 1686262505370, "duration": 637}
{"@timestamp": "2023-06-08T22:15:06.44Z", "cId": 70033, "startTs": 1686262505370, "duration": 732}
{"@timestamp": "2023-06-08T22:15:06.44Z", "cId": 70033, "startTs": 1686262505370, "duration": 734}
{"@timestamp": "2023-06-08T22:15:06.45Z", "cId": 40379, "startTs": 1686262505979, "duration": 158}
```


What's a few billion log events between friends? Perhaps this data is kept for 24 to 48 hours and is rolled up into summaries kept for years.

Pipeline Summaries per Customer

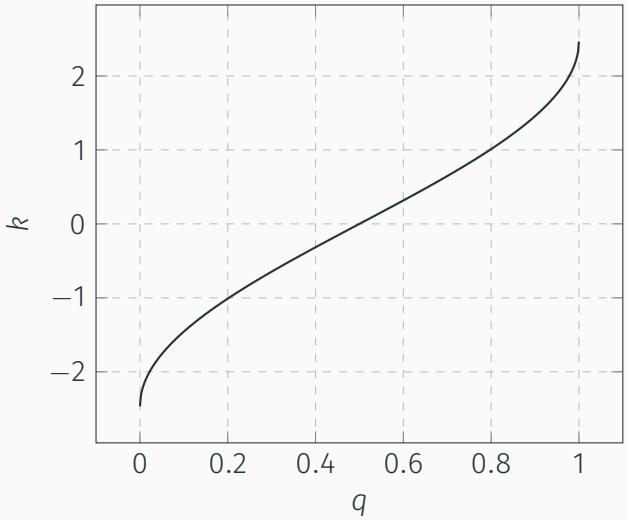
```
{"@timestamp": "2023-06-08T22:15:00Z", "cId": 10413, "n": 47794, "mu": 2396.42, "q50": 744.0, "q99": 3784.5}
{"@timestamp": "2023-06-08T22:15:00Z", "cId": 40529, "n": 3728, "mu": 826.01, "q50": 660.0, "q99": 5568.5}
{"@timestamp": "2023-06-08T22:15:00Z", "cId": 10275, "n": 15696, "mu": 2822.94, "q50": 663.0, "q99": 1724.5}
{"@timestamp": "2023-06-08T22:15:00Z", "cId": 40379, "n": 23377, "mu": 836.97, "q50": 651.0, "q99": 1511.0}
{"@timestamp": "2023-06-08T22:15:00Z", "cId": 70033, "n": 9927, "mu": 621.16, "q50": 542.0, "q99": 1073.0}
```

15 minute summaries *vastly* reduce the number of events and data storage required.

Wait, how do we aggregate this to build 7 and 30 day reports?

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

- 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% Ok, 20% Sucked

XXX: How bad did they suck?

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

Serialized T-Digests Per Customer

```
{
    "@timestamp": "2023-06-08T22:15:00Z",
    "cId": 481,
    "centroids": [
        [574, 1], [597, 1], [597, 1], [598, 1], [607, 1], [610, 1], [611, 1], [615, 1],
        [615, 1], [628, 1], [629, 1], [631, 1], [635, 1], [642, 1], [643, 1], [644, 1],
        [646, 1], [711, 1], [712, 1]
    ],
    "min": 574,
    "max": 712,
    "mean": 628.68,
    "q99": 711.5
}
```

Centroid length controled by δ, the compression constant.

Experiment 2 Results: 24 Hour $q(.99)$ Estimations from 15 Minute Rollups

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

Figure 4: Example of High Error Customer Distribution

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 NC-1701-D Image Credit: Paramount
- Google SRE: Four Golden Signals
- Dartmouth: The First and Second Derivatives
- Nyquist–Shannon sampling theorem
- Computing Extremely Accurate Quantiles Using t-Digests
- Sample Quantiles in Statistical Packages