

# OBSERVABILITY DATA ENGINEERING

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

---

Jack Neely

[jjneely@gmail.com](mailto:jjneely@gmail.com)

June 3, 2023

DevOps Observability Architect

# MONITORAMA PDX 2019: HOW TO KNOW IF SOMETHING IS “UP”

*What do I monitor?*

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

**Health** Pods are Running and Healthy

**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

Remember: There are FIVE lights!

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

## 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...

## 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. How heavily are you sampling your traces?

## 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. How heavily are you sampling your traces?

*Jack, we're an Enterprise!*

starship goes here



TRAFFIC

## 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

Most Accurate: Incremented in discrete whole numbers. Never misses an event.

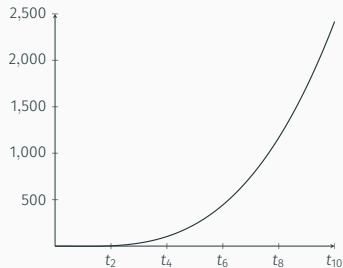
Synchronization Primitive: Allows for multiple observers.

Low Overhead: Easy implementation. No copying or recalling previous values.

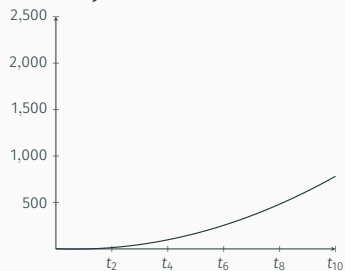
Fundamental: **Position!**

# REMEMBERING PHYSICS

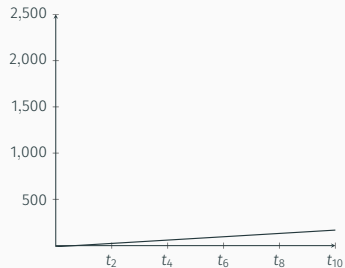
Position



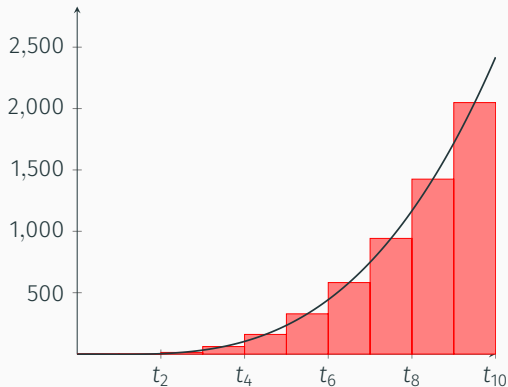
Velocity



Acceleration



# 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:

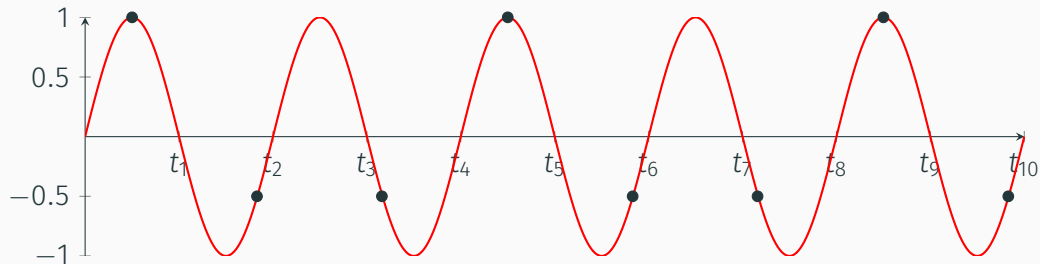
```
1 - (
  sum_over_time(
    sum without (status) (
      labels:http_server_requests:rate5m{
        status=~"5..", service="..."}[7d:5m]
    ) * 300 /
    sum_over_time(
      sum without (status) (
        labels:http_server_requests:rate5m{
          service="..."[7d:5m]
        ) * 300
      )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

# ERRORS

How do you measure CPU usage of a process?

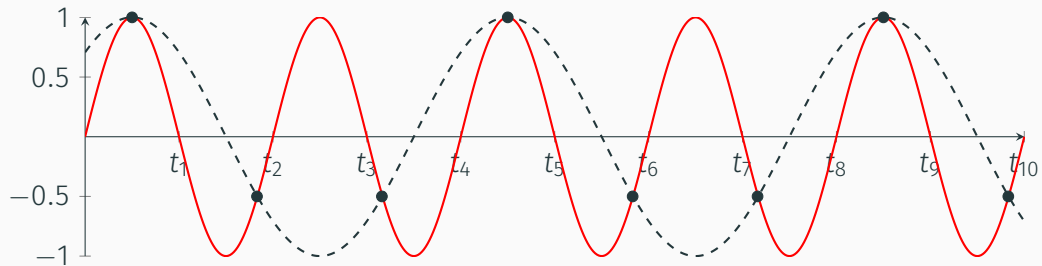
- a. Jiffies
- b. Percentages
- c. Seconds a Process is in the Running State
- d. All of the above

# NYQUIST-SHANNON SAMPLING THEOREM



*ScrapeInterval* >  $2f$

# NYQUIST-SHANNON SAMPLING THEOREM

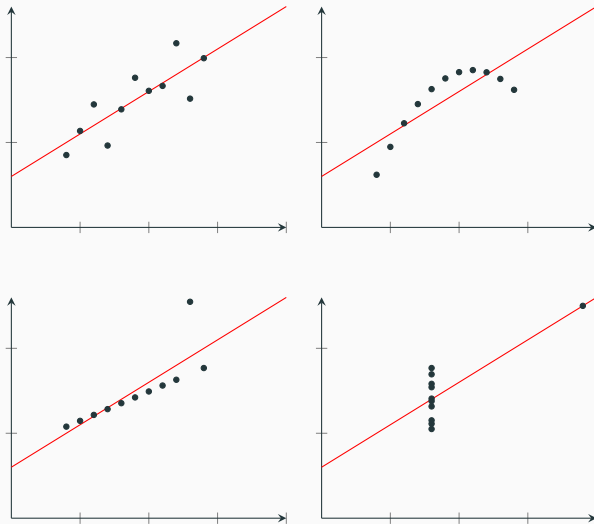


*ScrapeInterval* >  $2f$

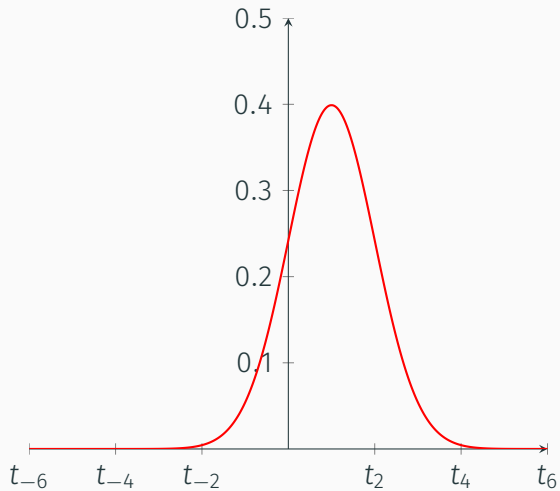


# LATENCY

# ANSCOMB'S QUARTET



# NONSTANDARD DISTRIBUTIONS



# SATURATION

CUSTOMERS