Observability Data Engineering

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

Jack Neely
jjneely@gmail.com

June 11, 2023

DevOps Observability Architect

As a DevOps Observability Architect...

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 cardinally 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 cardinally 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*

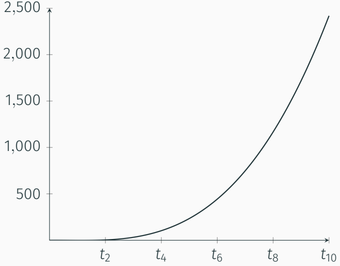
Most Accurate: Incremented in discrete whole numbers. Never misses an event.

Synchronization Primitive: Allows for multiple observers.

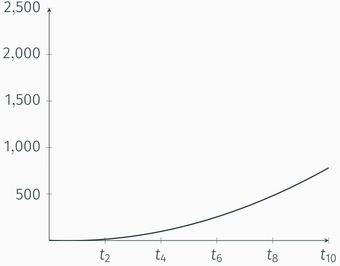
Low Overhead: Easy implementation. No copying or recalling previous values.

Fundamental: **Position!**

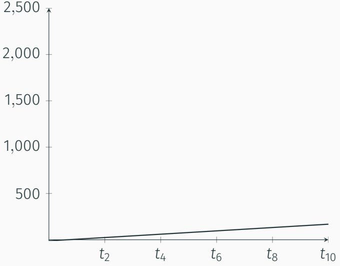
Position



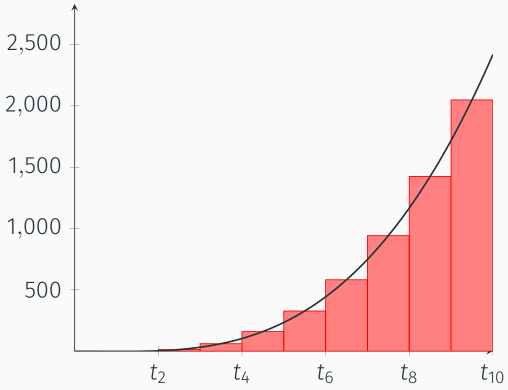
Velocity



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:

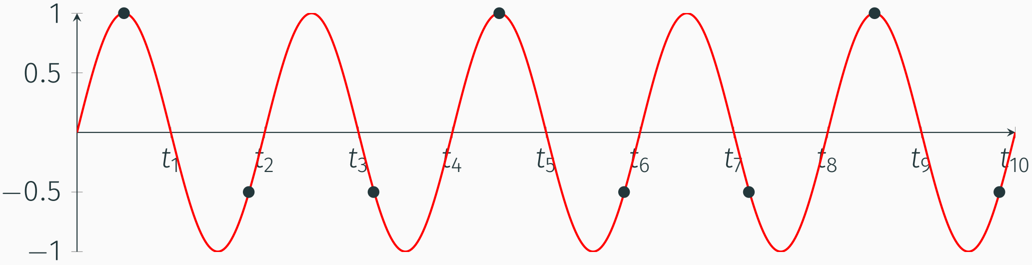
```
1 - (
  sum_over_time(
    sum without (status) (
      labels:http_server_requests:rate5m{
        status=~"5..", service="..."})[7d:5m]
  ) * 300 /
  sum_over_time(
    sum without (status) (
      labels:http_server_requests:rate5m{
        service="..."})[7d:5m]
  ) * 300
)
```

Errors

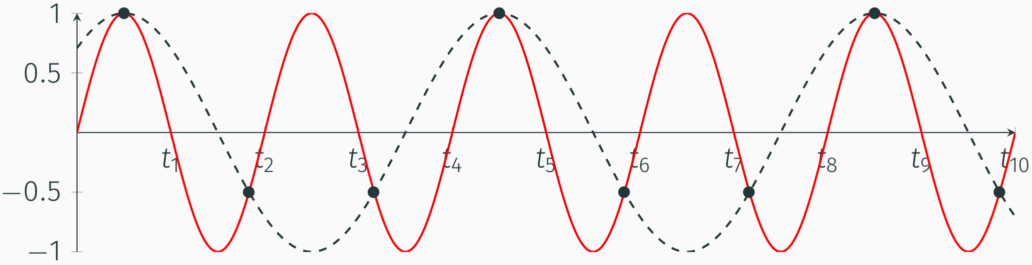
Measuring CPU Usage Over Time

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



$$ScrapeInterval > 2f$$

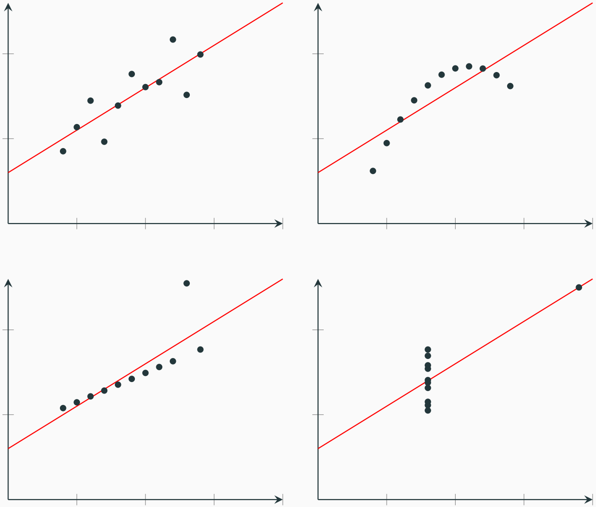


$$ScrapeInterval > 2f$$

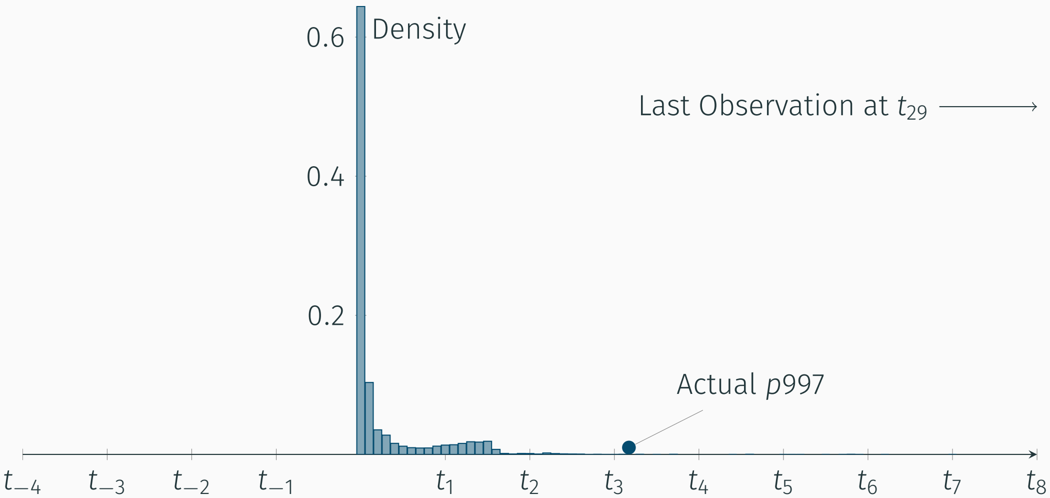
Latency

Anscomb's Quartet

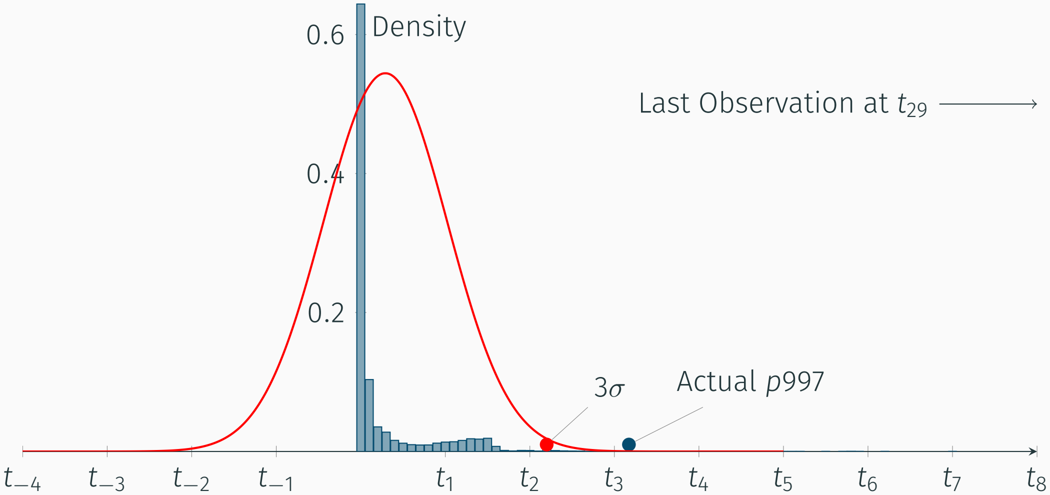
Summary Statistics	
N	11
$\mu\{x_1..x_n\}$	9.0
$\mu\{y_1..y_n\}$	7.5
$\sigma\{x_1..x_n\}$	3.16
$\sigma\{y_1..y_n\}$	1.94
r^2	0.67



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

Latency Key Takeaways

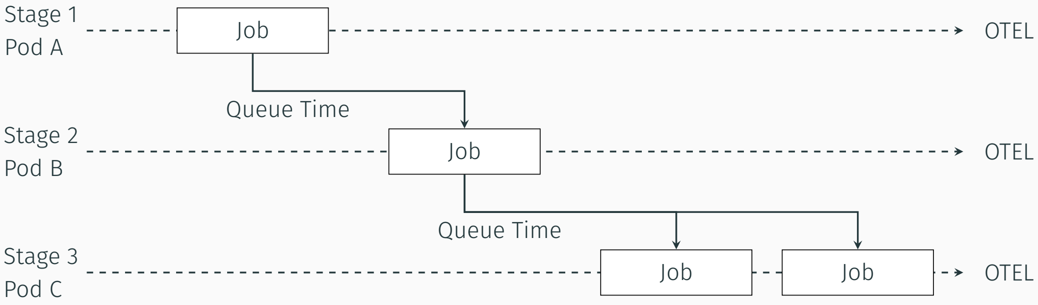
Question Averages

Use Quantiles to Represent Latency Spread

- Median or $q(0.50)$
- $q(0.90)$
- $q(0.95)$
- $q(0.99)$
- 3σ or $q(0.997)$
- Max or $q(1)$

Saturation

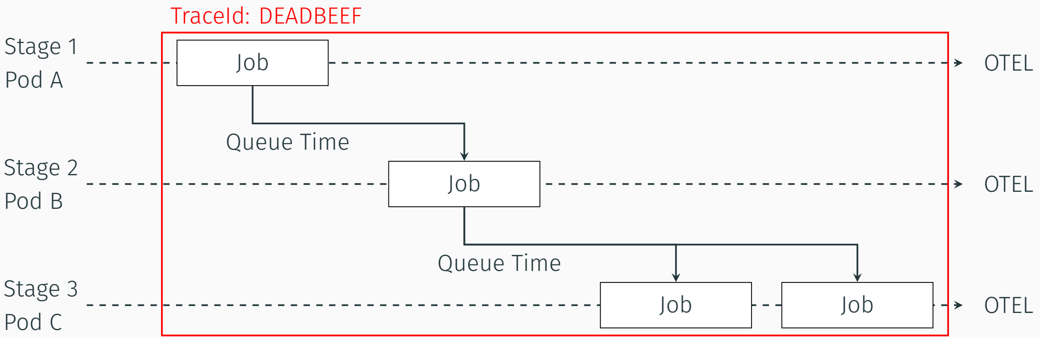
Tracing Pipelines



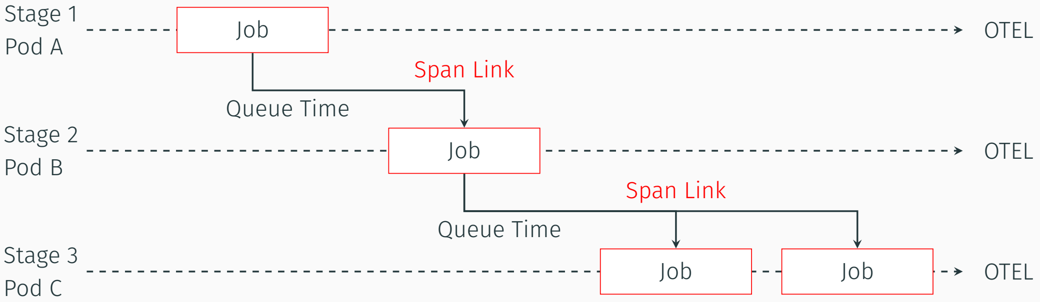
Freshness SLO X% of results are processed in Y time or less over the last Z days.

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.

```
{
    foo=bar,
    baz=sue
}
```

Health

Of Your Customers

FOOBAR