

# Beyond Line Charts

Why some diversity in telemetry visualization is long overdue.

Yao Yue, CEO @ IOP Systems

*An online version of these slides with interactive charts can be found [here](#).*

# About Me

Builder / Operator / Researcher

15+ years of large-scale systems experience;

7+ years as a pager-carrying oncall for a tier-1 service

Usually curious about *why*

professional viz-charts starer



# Reliability Engineer (2025 Edition)



What I think I do



What society thinks I do



What my boss thinks I do

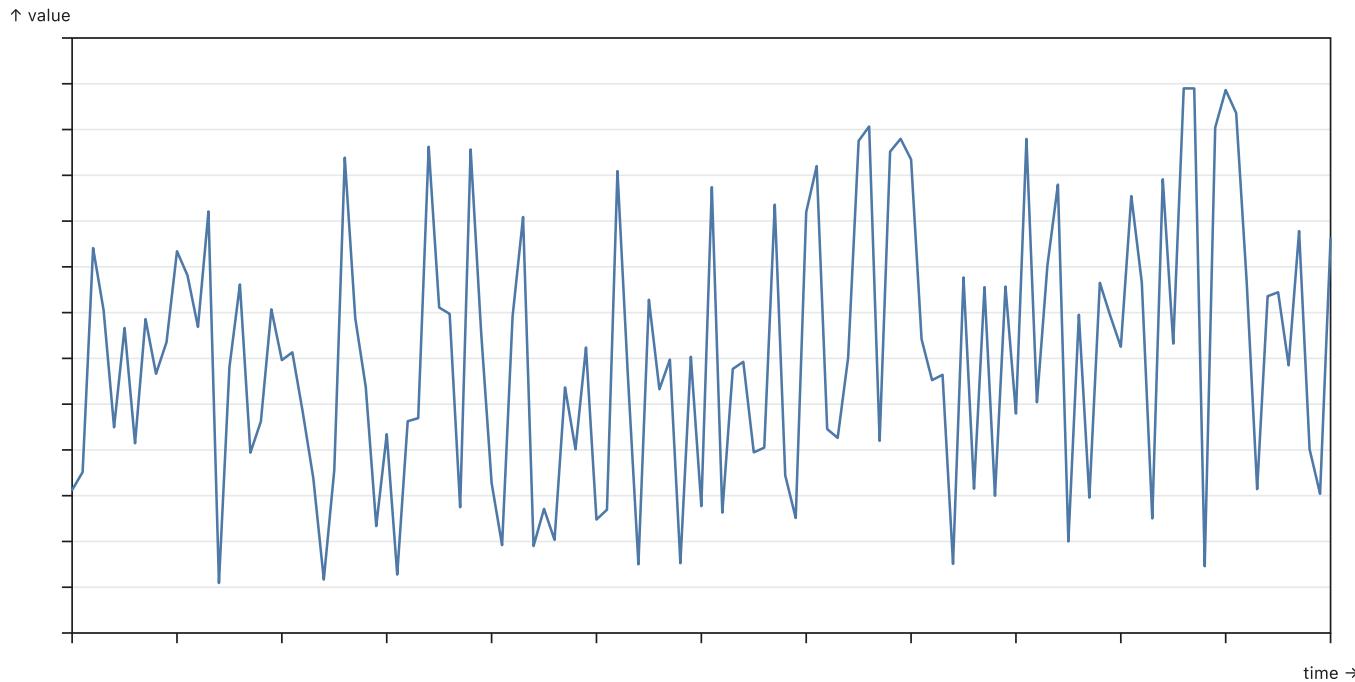


What I actually do

**“The most dangerous  
phrase in the language  
is ‘we’ve always done it  
this way.’”**

— *Grace Hopper*

# Value-over-time As A Line Chart

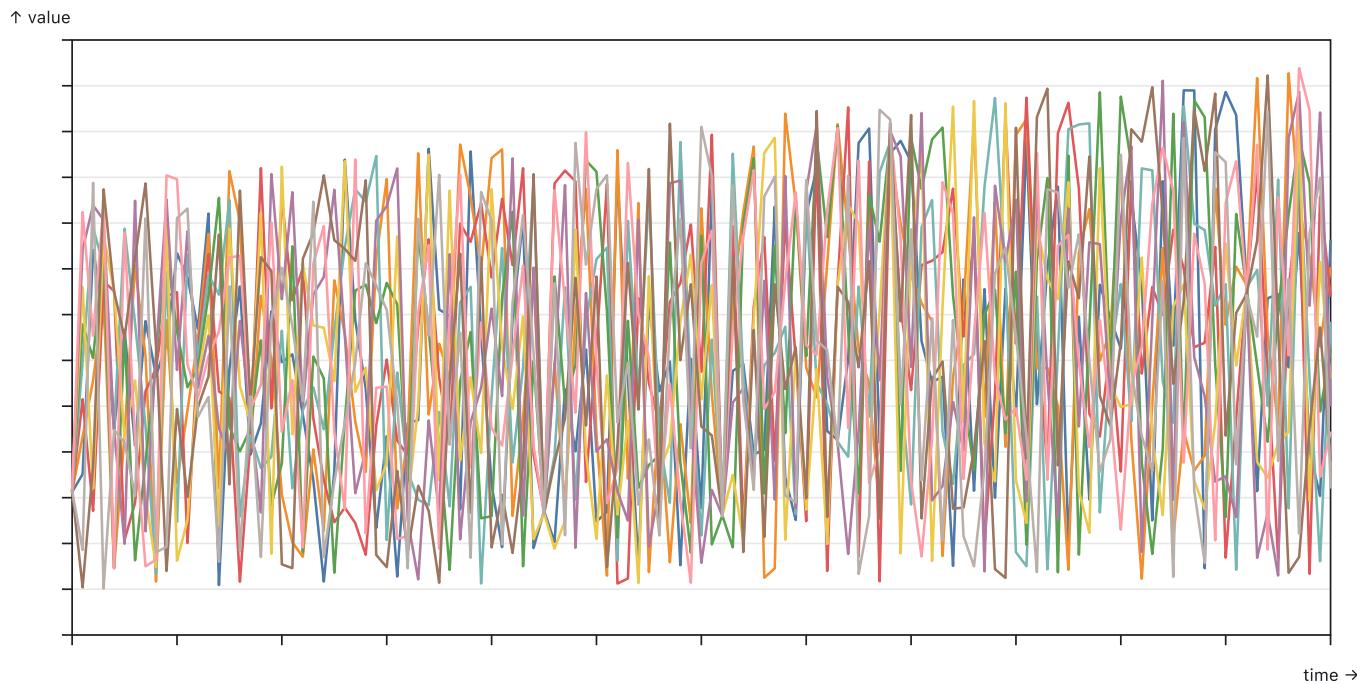


# What's A Line Chart?

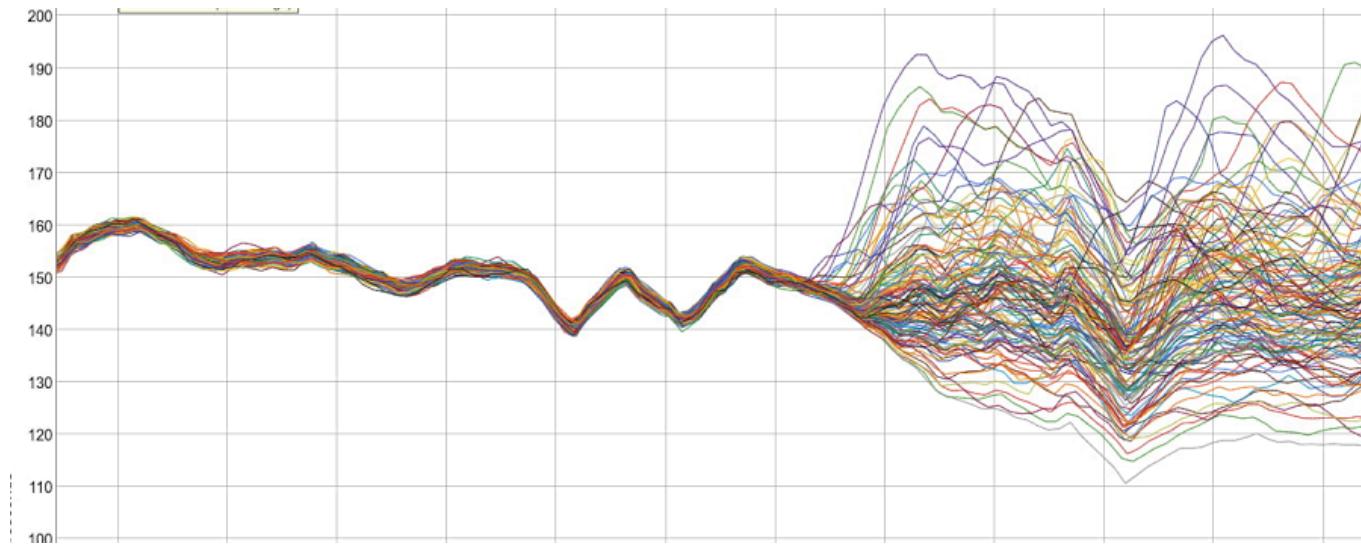
A type of chart that displays information as a series of data points connected by straight line segments.

The line segments, in particular, are a form of *interpolation*.

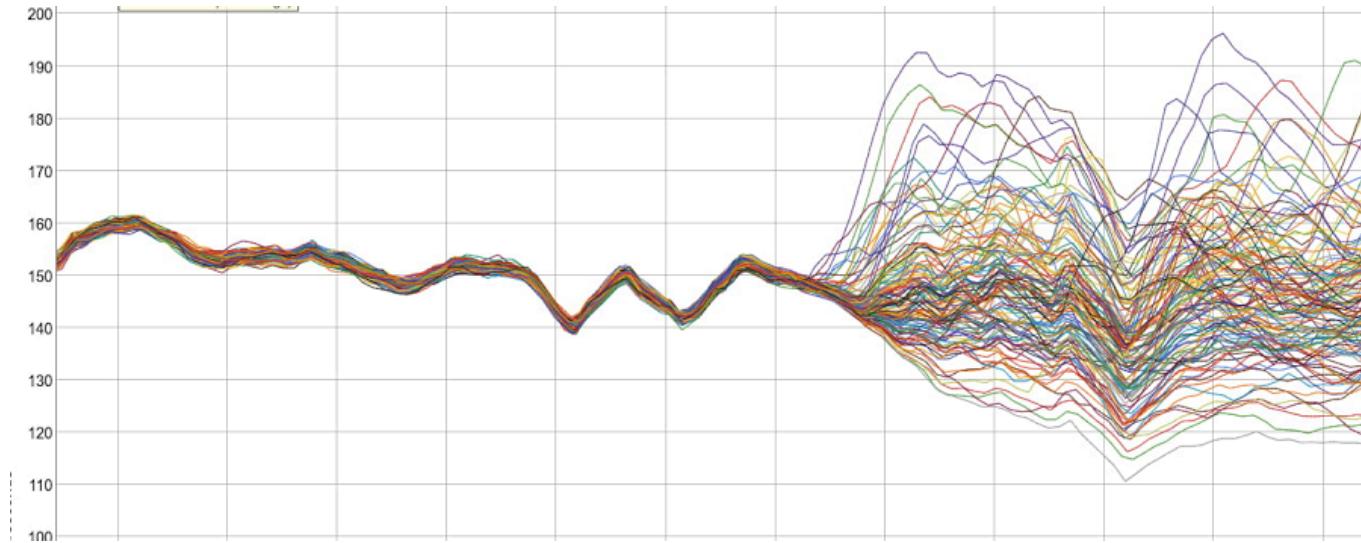
# Multi-line Madness



# Multi-line Madness



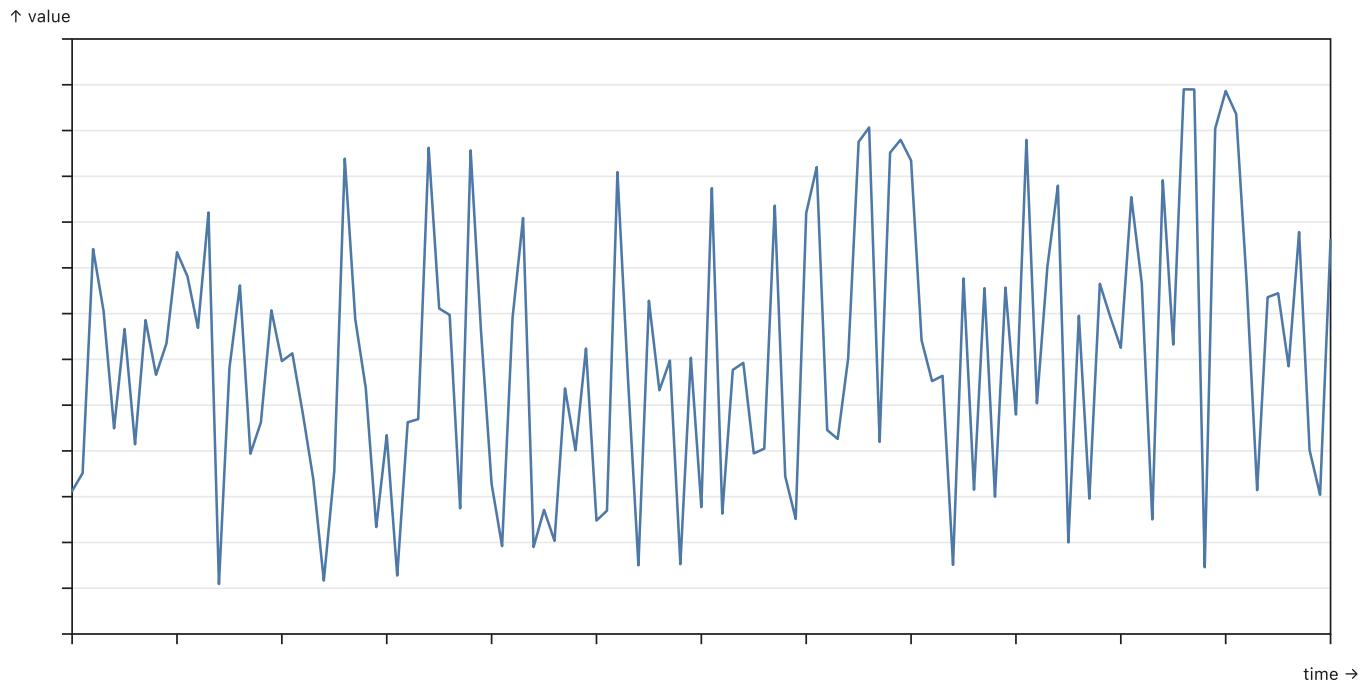
# Multi-line ~~Madness~~ Art



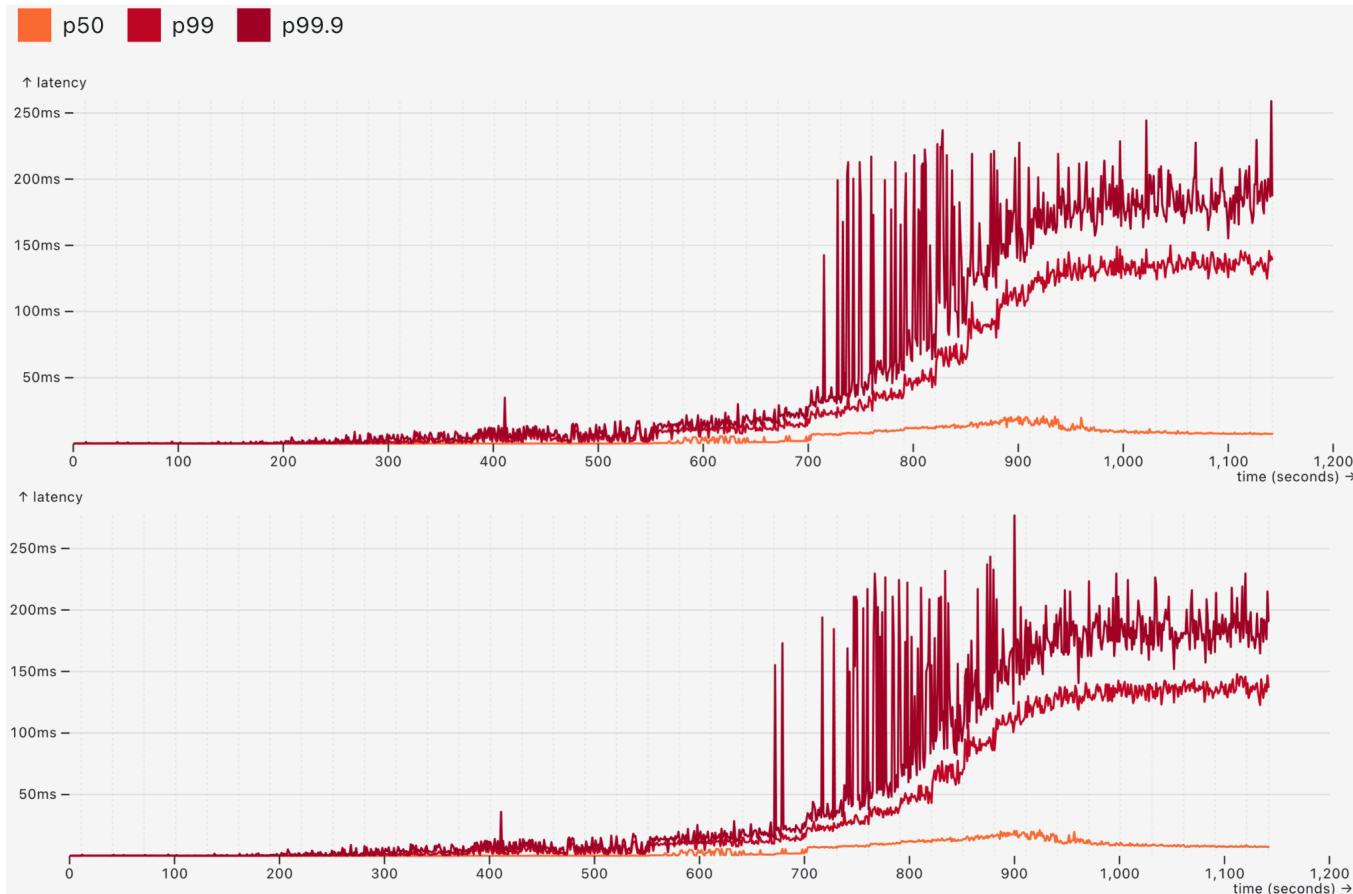
Museum of modern Borgman art

**Good data  
visualization makes  
the insight obvious**

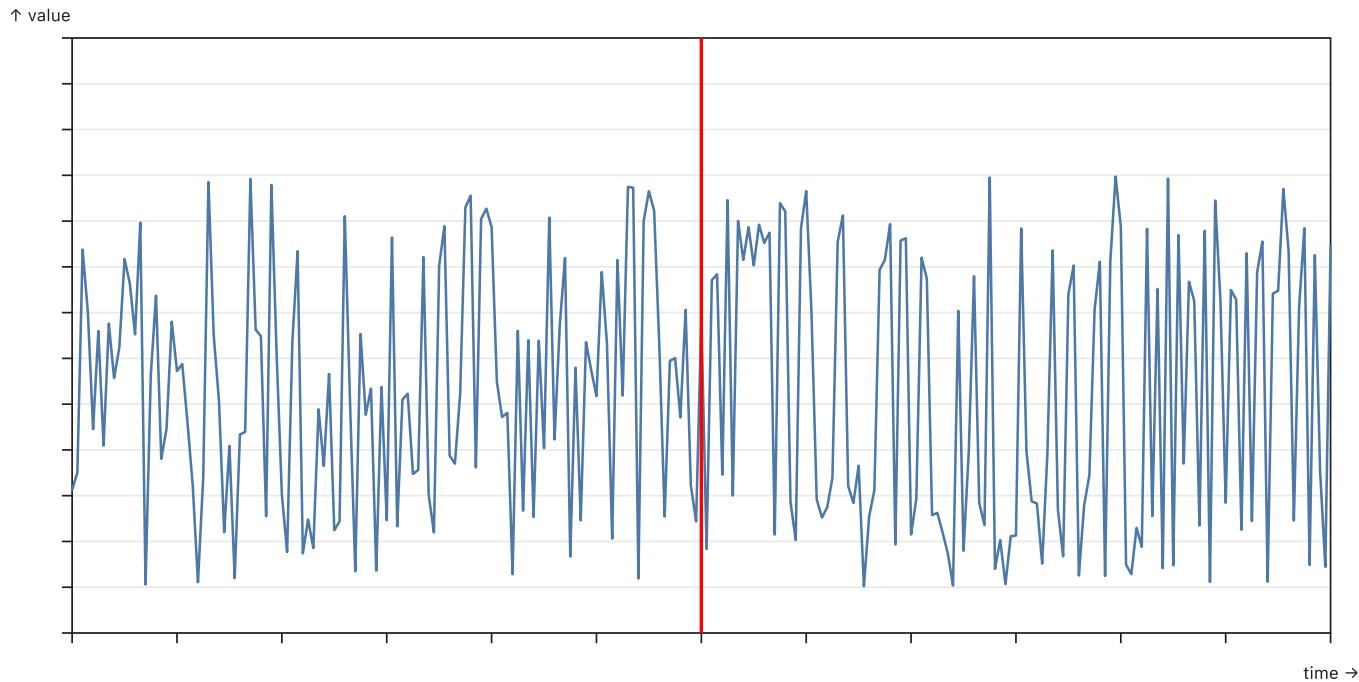
## *"IS THROUGHPUT CHANGING?"*



# "ARE THESE TWO LATENCY DISTRIBUTIONS MEANINGFULLY DIFFERENT?"



*"HAS P99 CHANGED AFTER THE DEPLOY?"*



# “Squinting-based Analysis”



# What's My Problem with Line Charts?

1. They can be distracting, because there are a lot of pixels—real, or interpolated.
2. They always imply “time”.

**Why are line charts  
synonymous with  
telemetry?**

# "Metrics must flow"—*writes* dominate TSDB

## Telemetry data as time series

- Most common operation is (real-time) append
- New time series creations are less common
- Reads are range based, but far sparser than writes

## Solidified over 30 years of database evolution

- Dedicated TSDB for the monitoring need of distributed services
- Append-friendly DBMSes such as DynamoDB, Cassandra are also designed for writes

# Time Series Storage Format

Metric+label  
based index

Time-centric  
value storage

Time Series Identifier

Time Series Samples

```
metric_name:  
label0 = value0,  
label1 = value1,  
...
```

time	value
t0	v0
t1	v1
t2	v2
...	...

# Visualizing Time Series Values

# Use Lines To Guide Flow of Time

# Only Keep the Line Marker

What is *better*?

**“Form follows  
function.”**

— *Louis Sullivan, architect*

# Telemetry

Data Shape

Telemetry Type

Operational Questions

# Telemetry

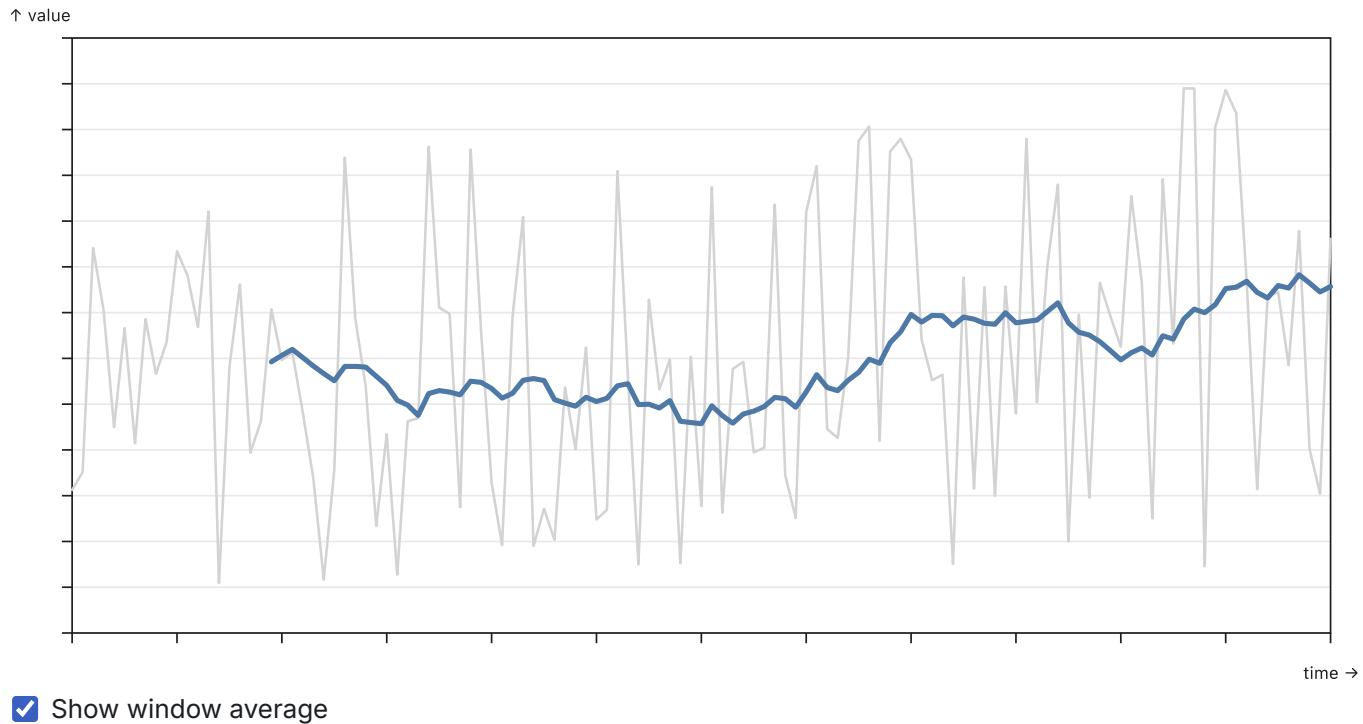
**Data Shape**

**Telemetry Type**

**Operational Questions**

Line charts are bad for  
busy data.

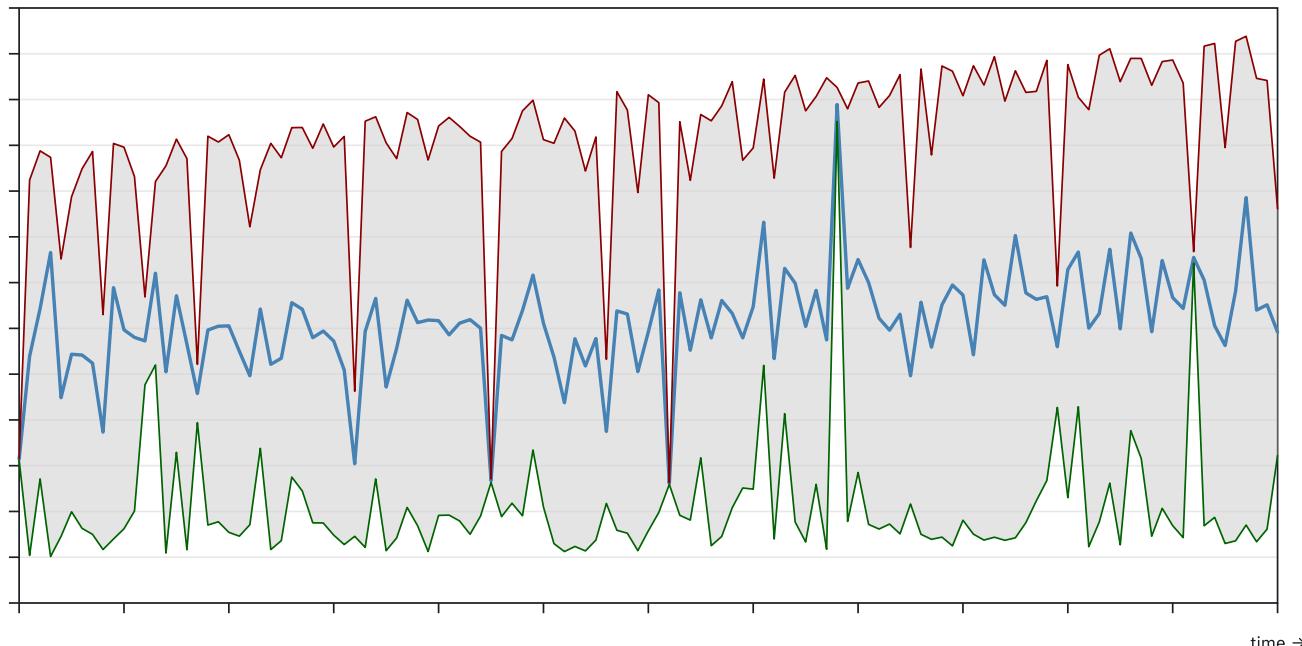
## *"IS THROUGHPUT CHANGING?"*



# "IS CLUSTER THROUGHPUT CHANGING?"

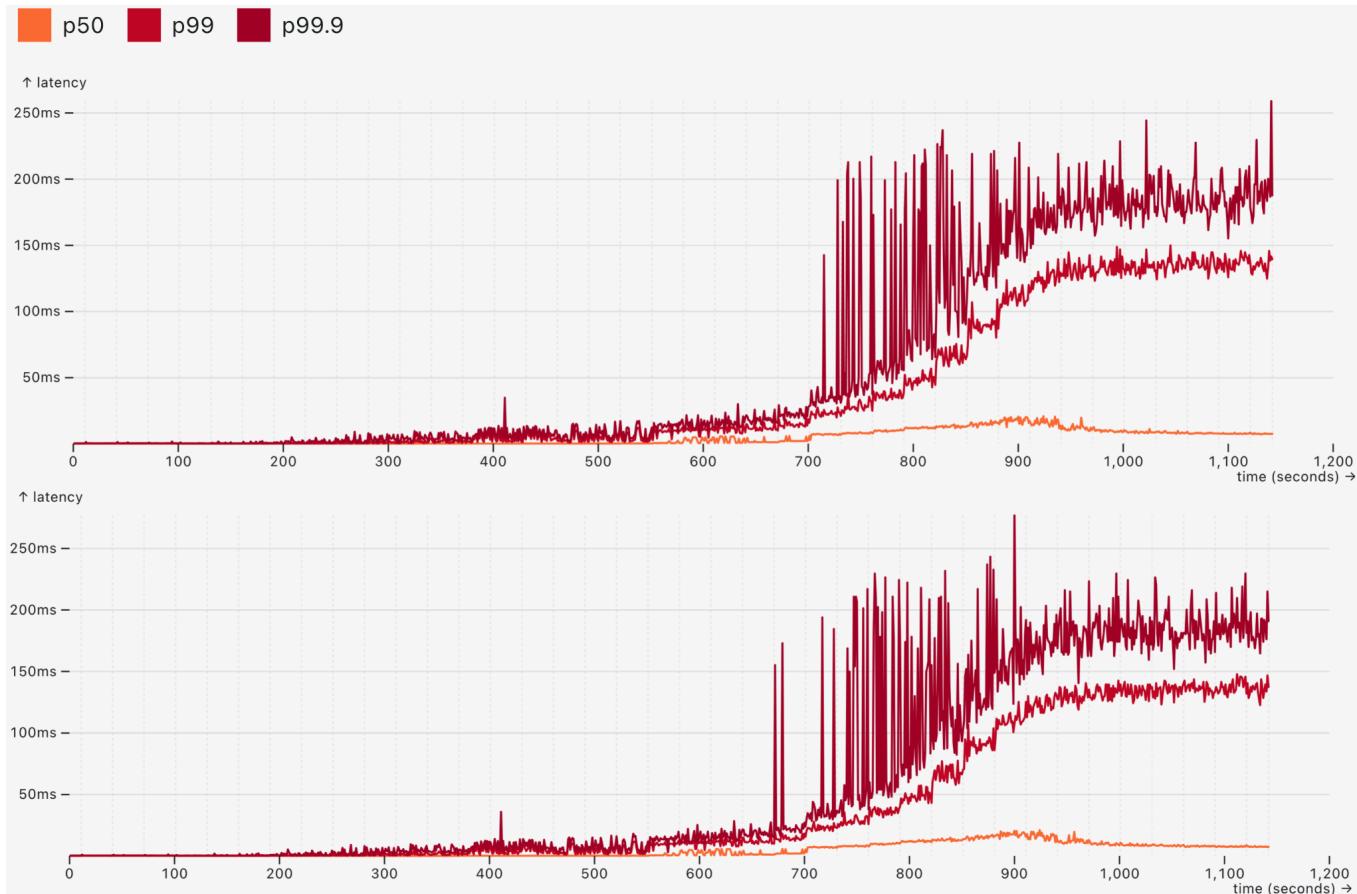
avg  max  min

↑ value

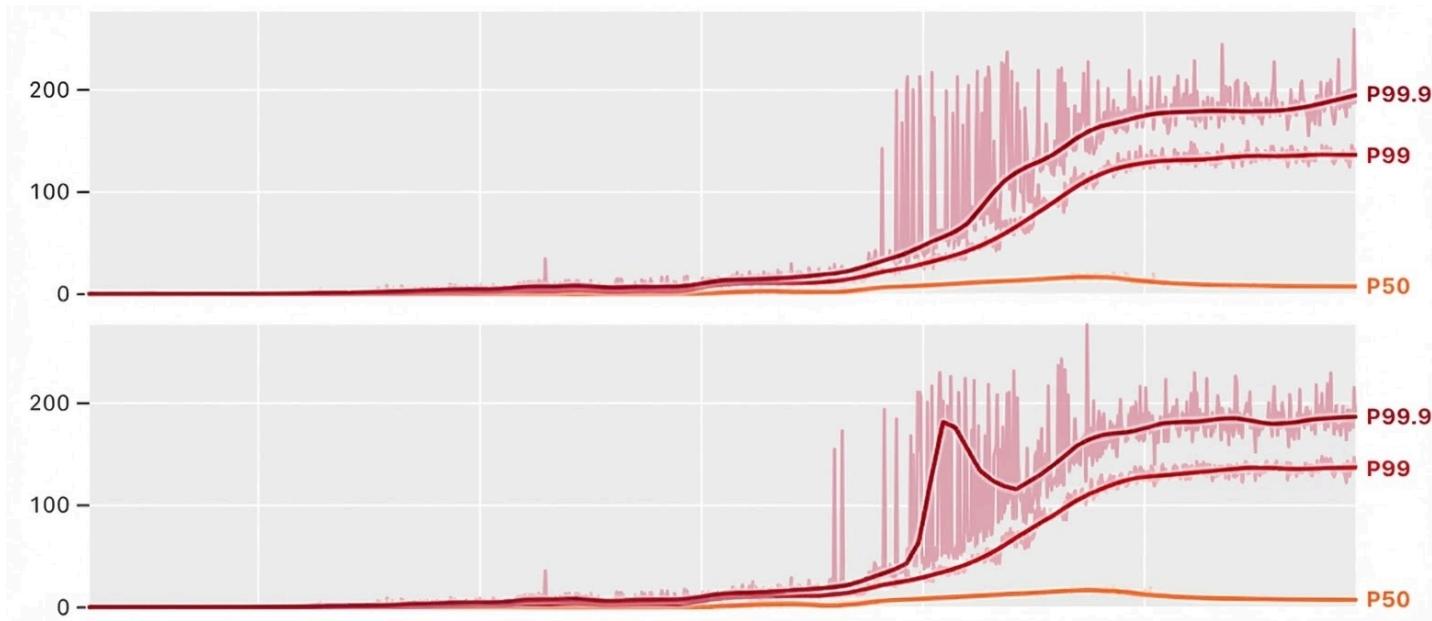


Show summary

# "ARE THESE TWO LATENCY DISTRIBUTIONS MEANINGFULLY DIFFERENT?"

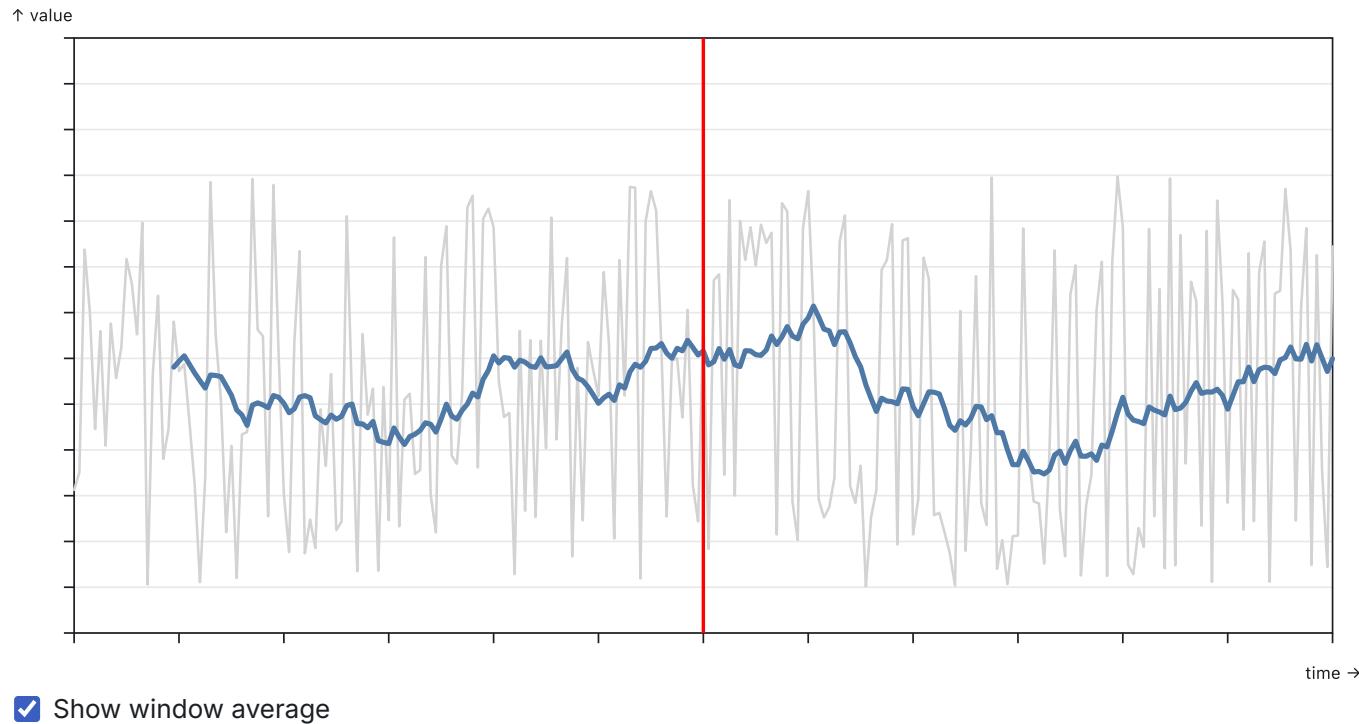


"ARE THESE TWO LATENCY DISTRIBUTIONS MEANINGFULLY DIFFERENT?"

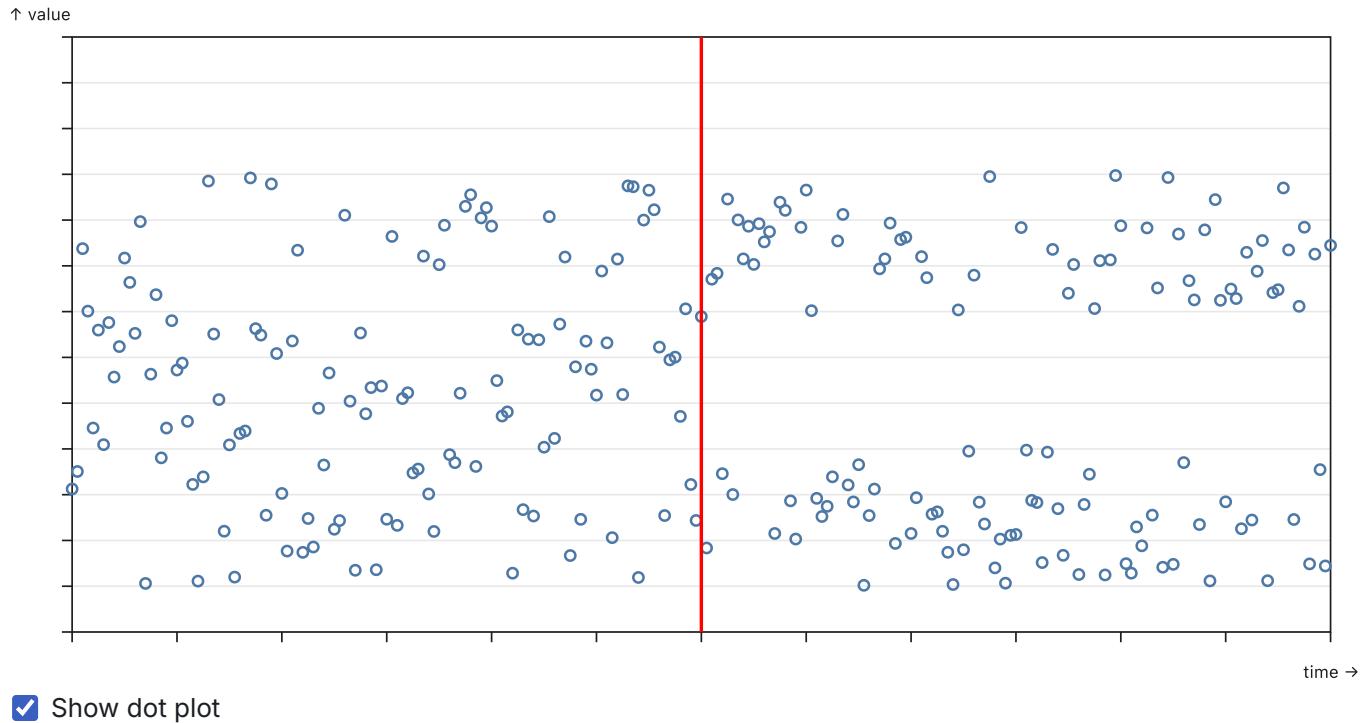


*Smoothed with LOWESS: an algorithm on local regression that's 50+ years old.*

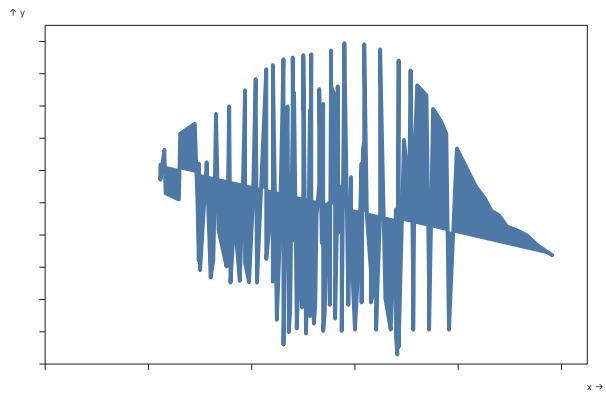
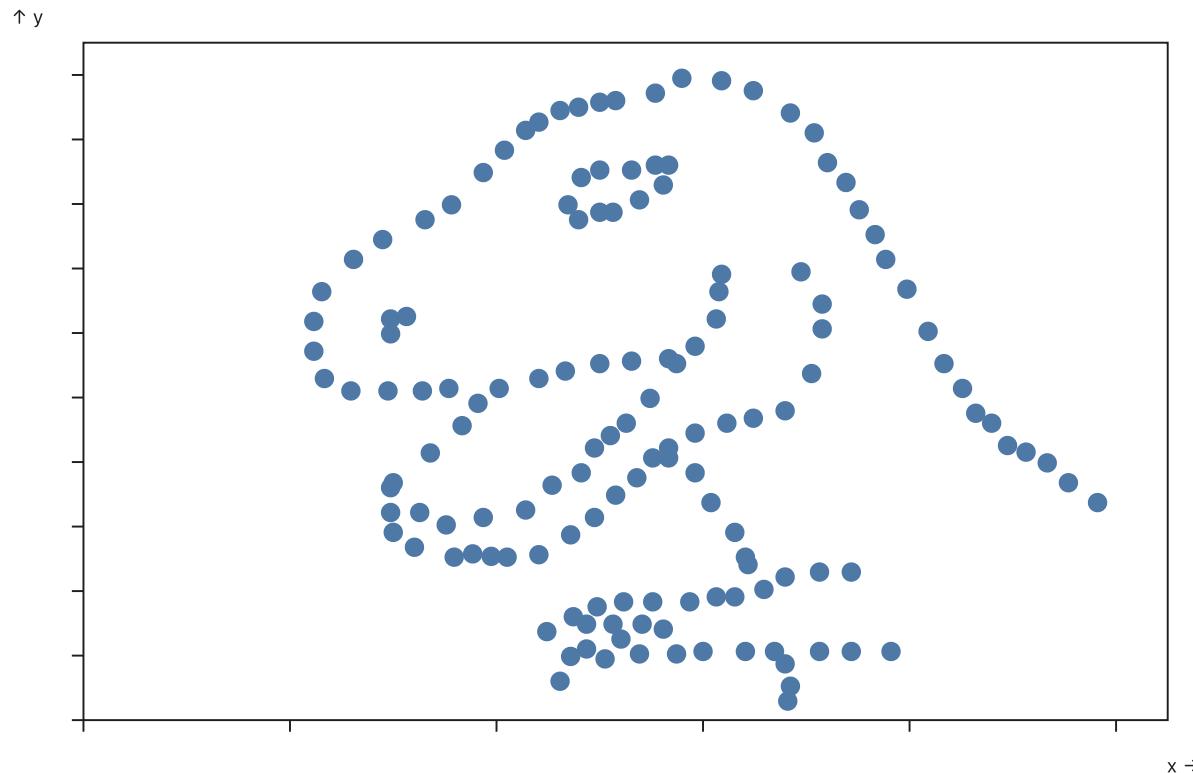
## *"HAS P99 CHANGED AFTER THE DEPLOY?"*



## *"HAS P99 CHANGED AFTER THE DEPLOY?"*



## The Power of Raw Data: See "Datasaurus"



The right visualization  
depends on the shape  
of data.

You have to *try*.

# Telemetry

Data Shape

Telemetry Type

Operational Questions

# Telemetry Type

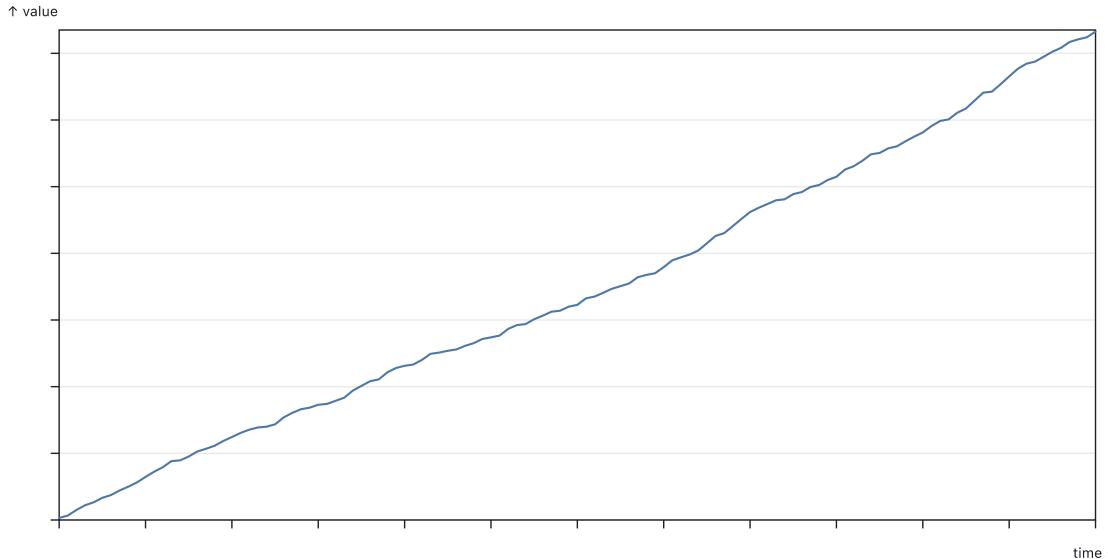
**Counter**: a cumulative measurement

**Gauge**: a spot measurement

**Histogram**: a distribution measurement

# Counters

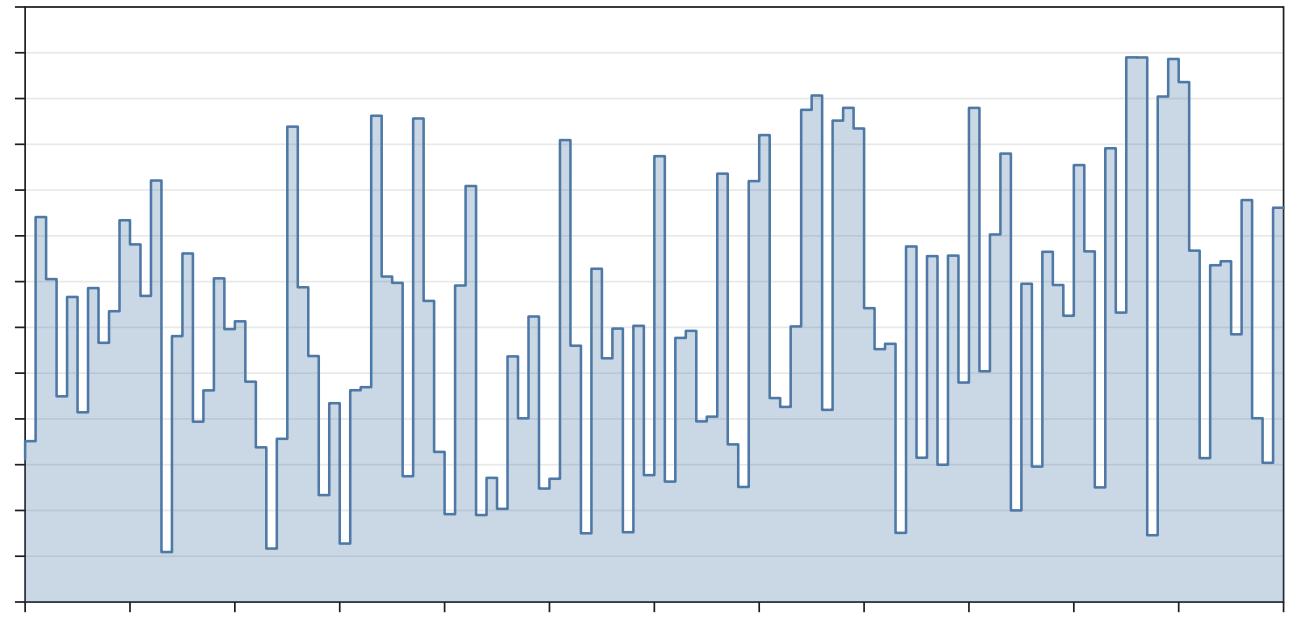
Interpolation between monotonic values with line segments? 



But! Almost always converted to delta counters for the slopes. 

# Delta Counters

↑ value

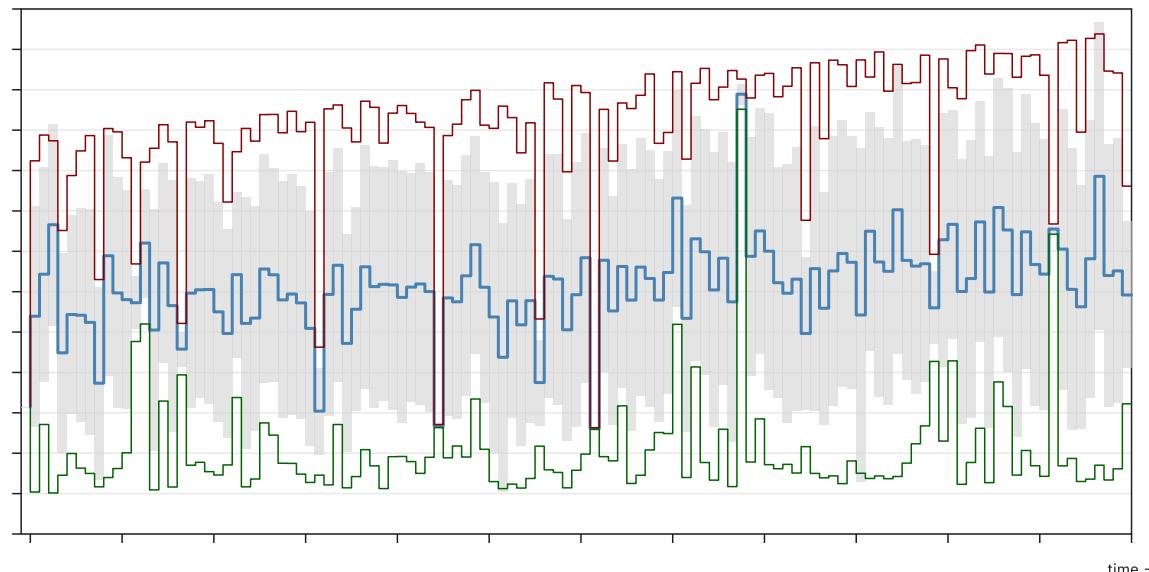


Counter style

# Multi-line Summary

avg    stddev    max    min

↑ value



time →

Counter style

Show stddev

# Telemetry Type

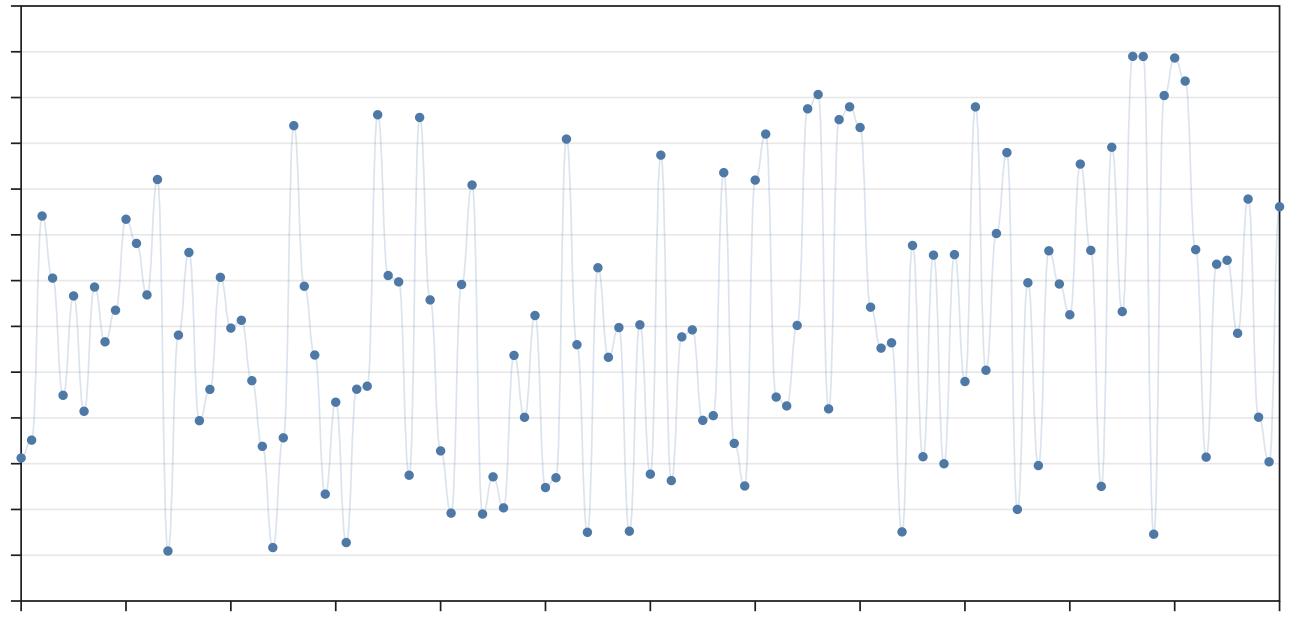
**Counter**: a cumulative measurement

**Gauge**: a spot measurement

**Histogram**: a distribution measurement

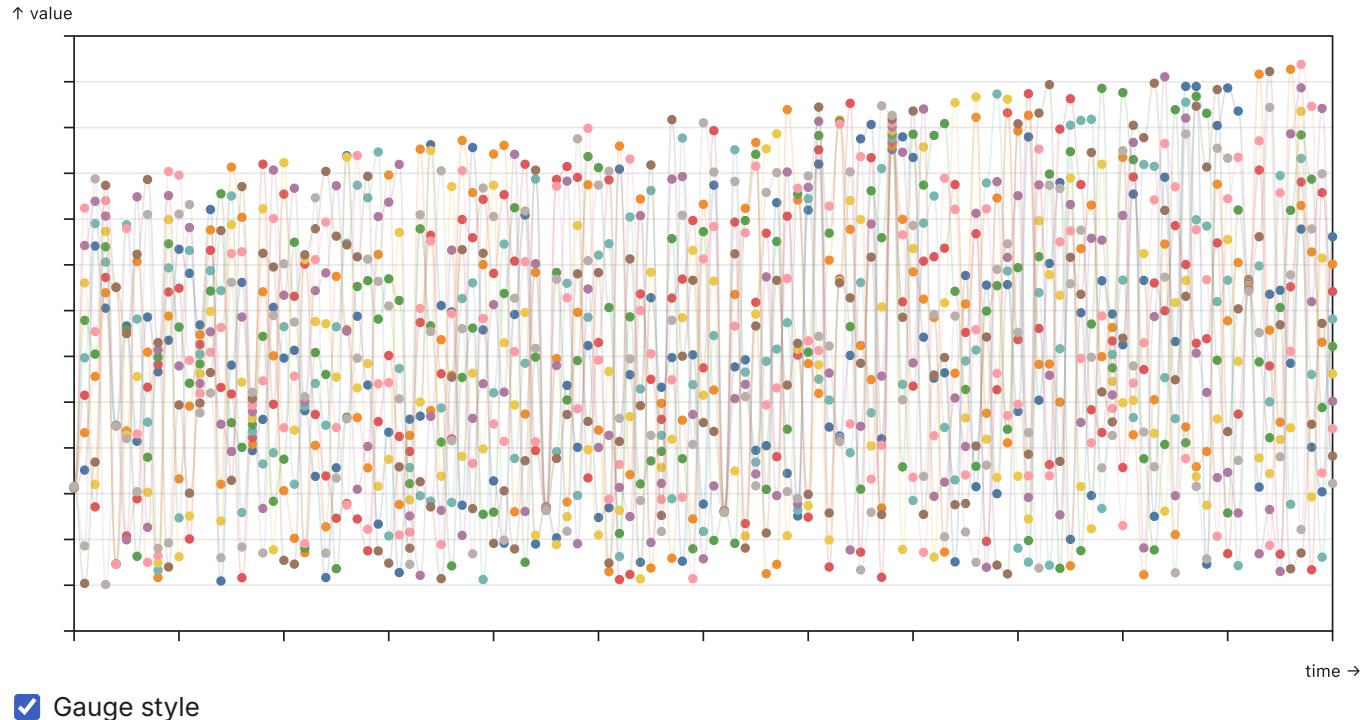
# Gauges: Points >>> Lines

↑ value



Gauge style

# Gauges: Points >> Lines



# Telemetry Type

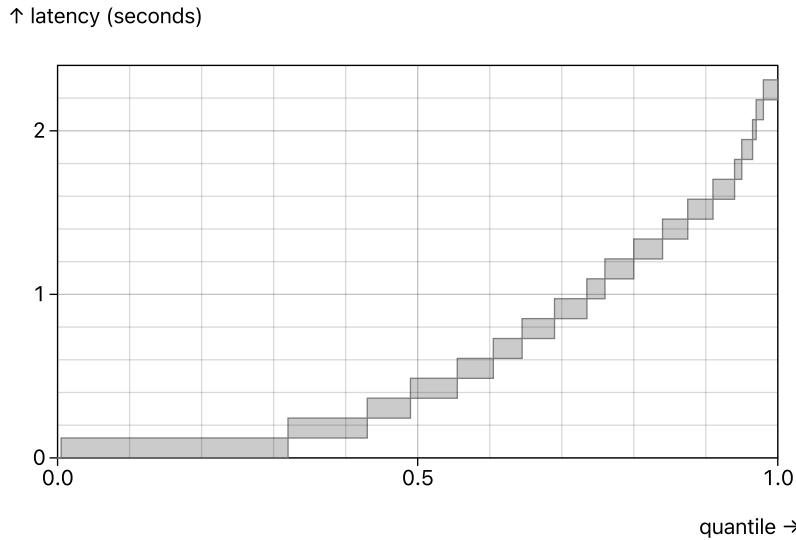
**Counter**: a cumulative measurement

**Gauge**: a spot measurement

**Histogram**: a distribution measurement

# Latency ❤ histogram

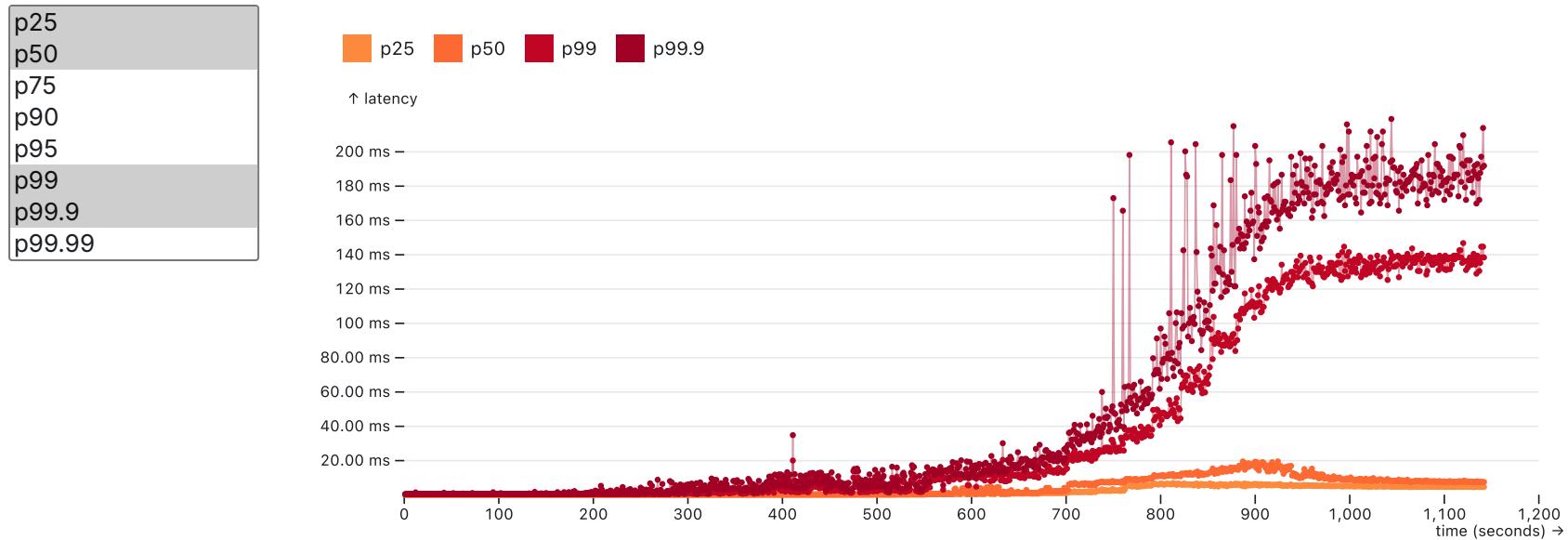
- A distribution: many quantiles, (approximate) values
- High dynamic range, high variance at tail



*Not just one number!*

# Histogram

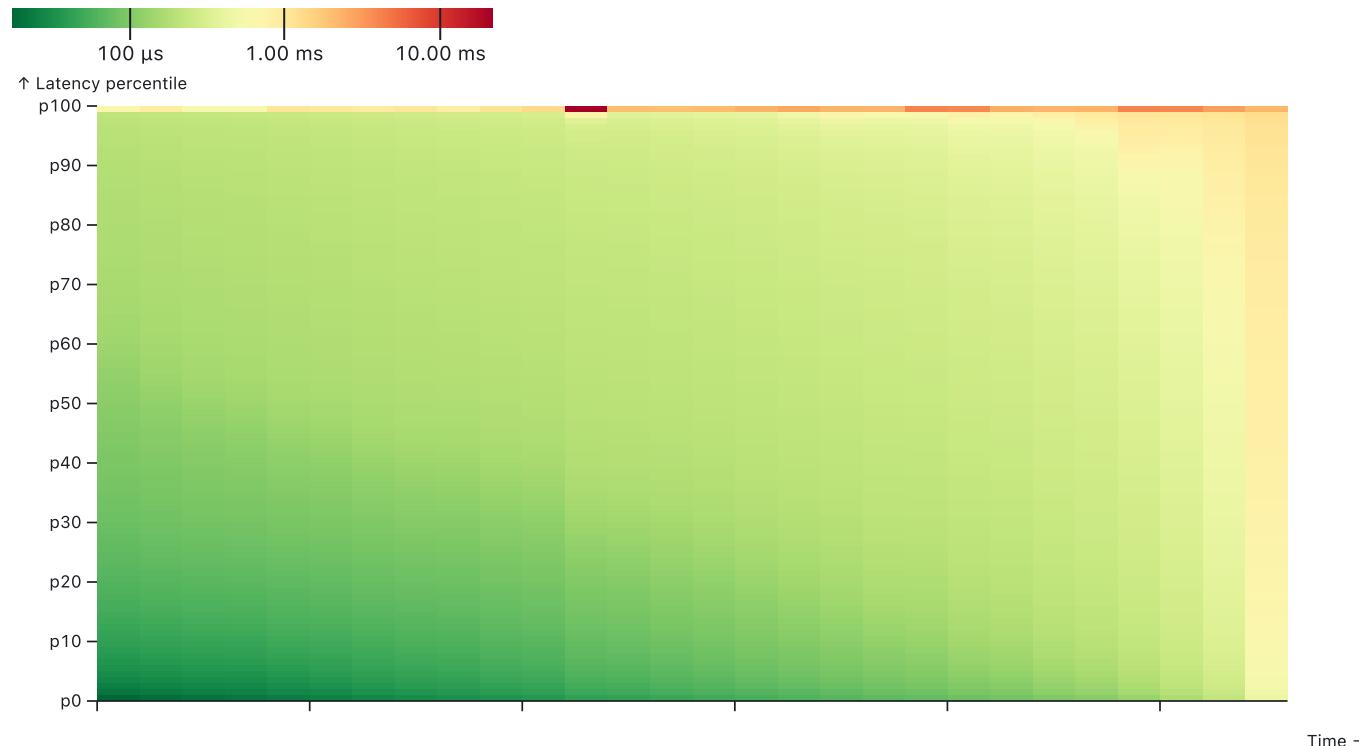
```
[{"quantile->"color", "value->"Y" }]
```



Gauge style

# Histogram

```
[(quantile→“Y”, value→“color”)]
```



The metric type offers  
strong hints on good  
visualization styles.

Make them the  
*default*.

# Telemetry

**Data Shape**

**Telemetry Type**

**Operational Questions**

# Some Common Questions

*"What happens to SLO if load goes up?"*

*"Has the new version regressed on throughput?"*

*"Which cloud SKU has the best ROI?"*

# Some Common Questions

*"What happens to SLO if **load** goes up?"*

*"Has the **new version** regressed on throughput?"*

*"Which **cloud SKU** has the best ROI?"*

# TSDB Storage Implication On Read

Group values by time ✓

Group values by labels ✓

Group values by other values ✗

metric\_name:  
label0 = value0,  
label1 = value1,  
...

time	value
t0	v0
t1	v1
t2	v2
...	...

# Some Common Questions

*"What happens to SLO if **load** goes up?"*

*"Has the new version regressed on throughput?"*

*"Which cloud SKU has the best ROI?"*

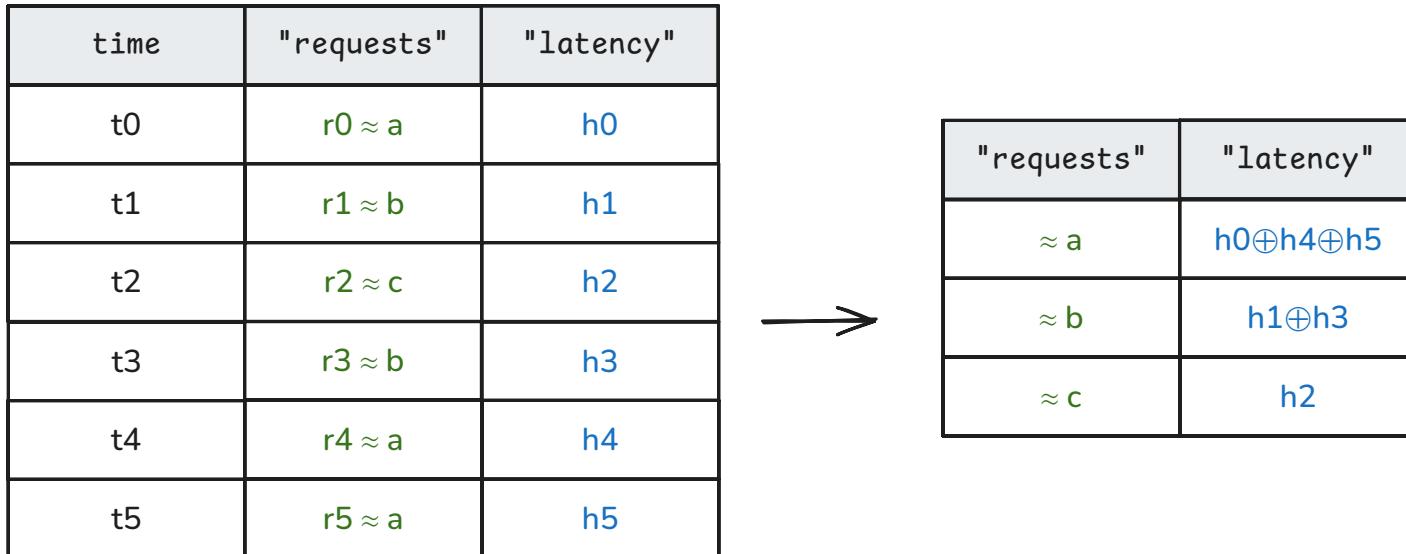
# Latency by Load: Gather Metrics

"requests":	
API = "GET",	
status = "200",	
...	
time	value
t0	r0
t1	r1
t2	r2
...	...

...

"latency/histo":	
API = "GET",	
status = "200",	
...	
time	value
t0	h0
t1	h1
t2	h2
...	...

# Latency by Load: Transform

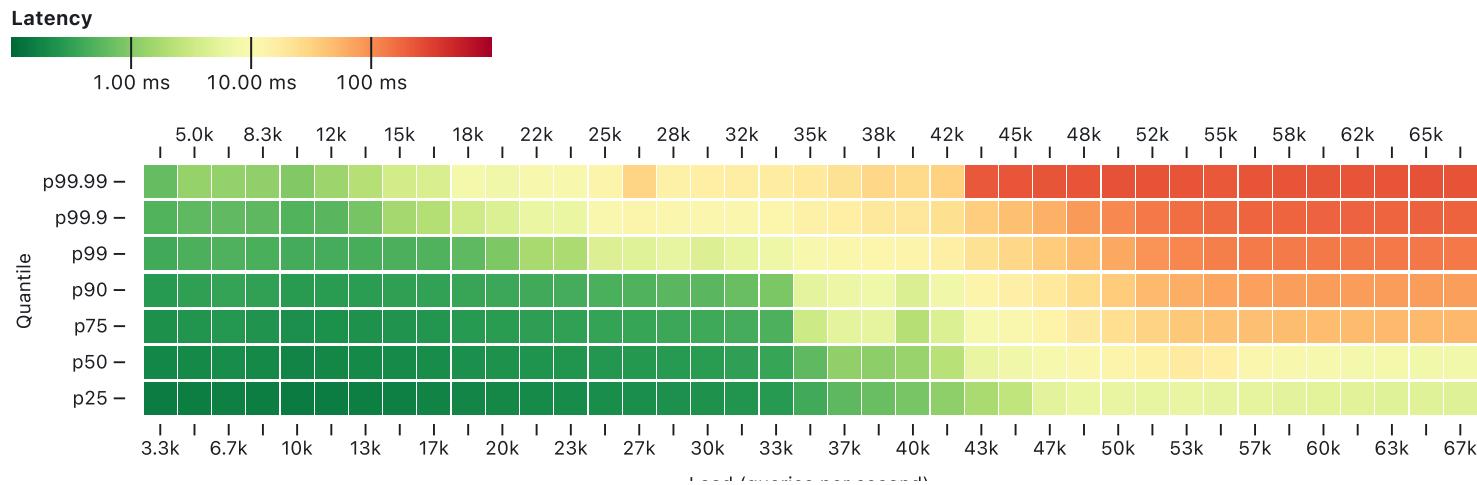


The diagram illustrates a transformation from a time-based table to a load-based table. An arrow points from the left table to the right table, indicating the mapping.

time	"requests"	"latency"
t0	$r0 \approx a$	$h0$
t1	$r1 \approx b$	$h1$
t2	$r2 \approx c$	$h2$
t3	$r3 \approx b$	$h3$
t4	$r4 \approx a$	$h4$
t5	$r5 \approx a$	$h5$

"requests"	"latency"
$\approx a$	$h0 \oplus h4 \oplus h5$
$\approx b$	$h1 \oplus h3$
$\approx c$	$h2$

# Load v.s. Latencies



# Some Common Questions

*"What happens to SLO if load goes up?"*

*"Has the **new version** regressed on throughput?"*

*"Which cloud SKU has the best ROI?"*

# Regression: Metrics & Attribute

<pre>"latency/histo":   API = "GET",   version = "0.0.1",   ...</pre>	
time	value
t0	h0
t1	h1
t5	h5
...	...

...

<pre>"latency/histo":   API = "GET",   version = "0.0.2",   ...</pre>	
time	value
t2	h2
t3	h3
t4	h4
...	...

# Regression: Transform

time	"version"	"latency"
t0	"0.0.1"	h0
t1	"0.0.1"	h1
t2	"0.0.2"	h2
t3	"0.0.2"	h3
t4	"0.0.2"	h4
t5	"0.0.1"	h5



"version"	"latency"
"0.0.1"	$h0 \oplus h1 \oplus h5$
"0.0.2"	$h2 \oplus h3 \oplus h4$

# Regression A/B Analysis

Throughput (kQPS)

250



■ A ■ B

↑ Latency percentile

100

90

80

70

60

50

40

30

20

10

0

0

0.0625

0.125

0.25

0.5

1

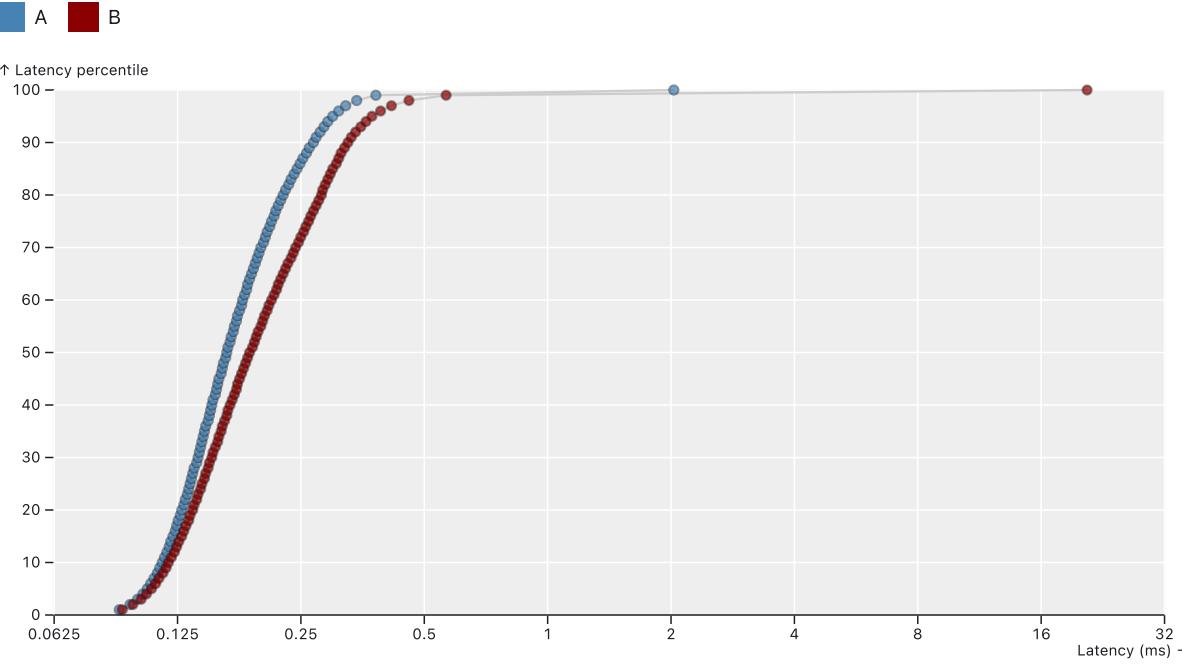
4

8

16

32

Latency (ms) →



# Some Common Questions

*"What happens to SLO if load goes up?"*

*"Has the new version regressed on throughput?"*

*"Which **cloud SKU** has the best ROI?"*

# SKU ROI: Gather Metrics & Prices

External Data

SKU	Instance Type	Hourly Price
g6e.xlarge	L40S GPU (NVIDIA Ada)	\$0.6840
g5g.xlarge	ARM + T4G GPU	\$0.1243
g5.xlarge	A10G GPU (NVIDIA Ampere)	\$0.5213
c7i.xlarge	Intel (Gen 7)	\$0.0823
c6i.xlarge	Intel (Gen 6)	\$0.0876
c7a.xlarge	AMD (Gen 7)	\$0.1066
c6a.xlarge	AMD (Gen 6)	\$0.0572
c8g.xlarge	ARM Graviton4	\$0.0731
c7g.xlarge	ARM Graviton3E	\$0.0801
c6g.xlarge	ARM Graviton3	\$0.0652

"tokens":  
sku="c8g.xlarge"  
...

time	value
t0	r0
t1	r1
t2	r2
...	...

...

"tokens":  
sku="g5g.xlarge"  
...

time	value
t0	r0
t1	r1
t2	r2
...	...

# SKU ROI: Transform



The diagram illustrates a data transformation process. On the left, there are two tables. The first table, titled "time", has columns "time", "sku", and "tokens". It contains three rows with "time" values "t0" and "sku" values "g5g.xlarge", "g6e.xlarge", and "g5.xlarge", all associated with "tokens" value "r0". The second table, titled "price", has columns "sku" and "price". It contains six rows with "sku" values "g5g.xlarge", "g6e.xlarge", "g5.xlarge", "c8g.xlarge", "c7a.xlarge", and "c7i.xlarge", and "price" values "\$0.1243", "\$0.6840", "\$0.5213", "\$0.0731", "\$0.1066", and "\$0.0823" respectively. An arrow points from these two tables to the right, where a third table is shown. This target table has columns "tokens", "sku", and "price". It contains six rows, each mapping a "tokens" value ("r0" in all cases) to a "sku" value and its corresponding "price". The "sku" values are color-coded to match the "sku" values in the second table: "g5g.xlarge" (red), "g6e.xlarge" (purple), "g5.xlarge" (orange), "c8g.xlarge" (blue), "c7a.xlarge" (green), and "c7i.xlarge" (teal). The "price" values are also color-coded: "\$0.1243" (red), "\$0.6840" (purple), "\$0.5213" (orange), "\$0.0731" (blue), "\$0.1066" (green), and "\$0.0823" (teal).

time	"sku"	"tokens"
t0	"g5g.xlarge"	r0
t0	"g6e.xlarge"	r0
t0	"g5.xlarge"	r0

"sku"	"price"
"g5g.xlarge"	\$0.1243
"g6e.xlarge"	\$0.6840
"g5.xlarge"	\$0.5213
"c8g.xlarge"	\$0.0731
"c7a.xlarge"	\$0.1066
"c7i.xlarge"	\$0.0823

"tokens"	"sku"	"price"
r0	"g5g.xlarge"	\$0.1243
r0	"g6e.xlarge"	\$0.6840
r0	"g5.xlarge"	\$0.5213
r0	"c8g.xlarge"	\$0.0731
r0	"c7a.xlarge"	\$0.1066
r0	"c7i.xlarge"	\$0.0823

# LLM Inference Cost per Token

Cost per 1M tokens (\$)



	135M SmolLM	360M SmolLM	1B Llama 3.2	1.7B SmolLM	2B Gemma 2	3B Llama 3.2
g6e.xlarge	1.0	1.2	1.0	1.3	2.1	2.0
g5g.xlarge	0.4	0.5	0.7	0.8	1.5	1.5
c7i.xlarge	0.4	1.0	2.2	2.8	4.7	5.3
c6i.xlarge	0.4	0.8	1.8	2.5	4.2	4.4
c6a.xlarge	0.2	0.5	1.2	1.6	2.8	2.8
c7a.xlarge	0.2	0.4	1.0	1.2	2.3	2.3
c6g.xlarge	0.2	0.3	0.9	1.1	2.2	2.1
c8g.xlarge	0.1	0.2	0.7	0.8	1.7	1.6
c7g.xlarge	0.1	0.2	0.7	0.8	1.6	1.6
g5.xlarge	0.8	0.9	0.7	1.0	1.6	1.5

# Telemetry As Panel Data

Does Pandas  
ring a bell?

time	"throughput"	"p99"
t0	10	127
t1	6	88
t2	25	385
t3	14	101
t4	18	76
t5	3	92
t0	7	64
t1	19	123
t2	12	256

...

"source"	"version"
"hostA"	"0.0.1"
"hostA"	"0.0.1"
"hostA"	"0.0.1"
"hostA"	"0.0.2"
"hostA"	"0.0.2"
"hostA"	"0.0.2"
"hostB"	"0.0.1"
"hostB"	"0.0.1"
"hostB"	"0.0.1"

# Telemetry As Panel Data

time	"throughput"	"p99"
t0	10	127
t1	6	88
t2	25	385
t3	14	101
t4	18	76
t5	3	92
t0	7	64
t1	19	123
t2	12	256

...

"source"	"version"
"hostA"	"0.0.1"
"hostA"	"0.0.1"
"hostA"	"0.0.1"
"hostA"	"0.0.2"
"hostA"	"0.0.2"
"hostA"	"0.0.2"
"hostB"	"0.0.1"
"hostB"	"0.0.1"
"hostB"	"0.0.1"

**Timeless questions  
deserve timeless  
schemas.**

# Call for Action

# To Practitioners

**Telemetry data have life beyond your TSDB!**

1. Pay attention to data type, shape
2. Learn about viz “classics” and general statistics
3. Transform data to give direct answers

# To Observability Providers

1. Better design defaults by telemetry type;
2. Enable basic statistical summary via UI;
3. Design for alternative data schema; and connect with broader analytical tools

# Questions?

Meanwhile, enjoy  
a viz gallery by  
IOP Systems—

