# Data Rules for Observability

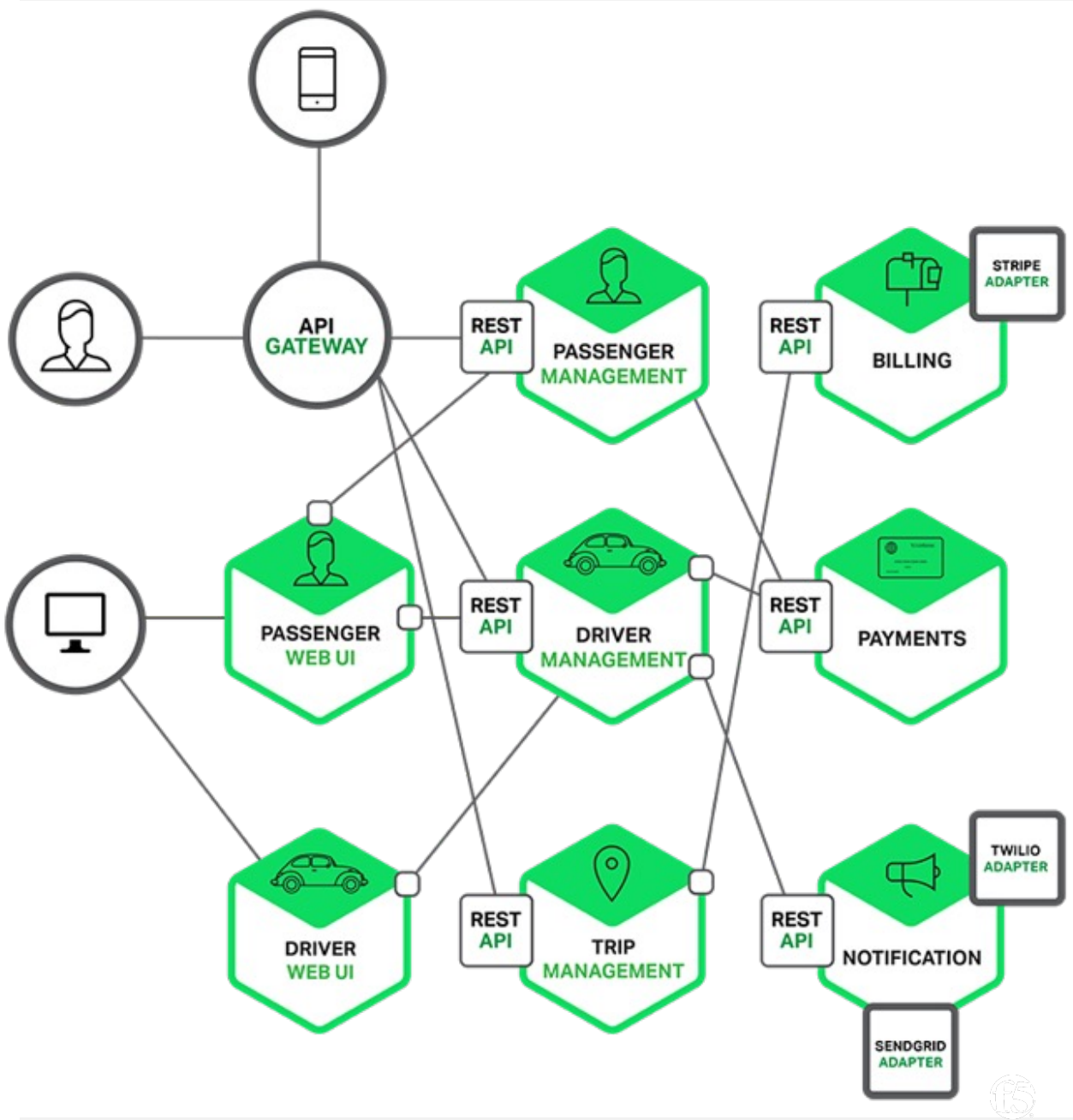
Dave McAllister - NGINX

# So Why Observability?

# Microservices!

Single application composed of many loosely coupled and independently deployable smaller services

- Often polyglot in nature
- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Often in Cloud environments
- Organized around business capabilities
- Each potentially owned by a small team

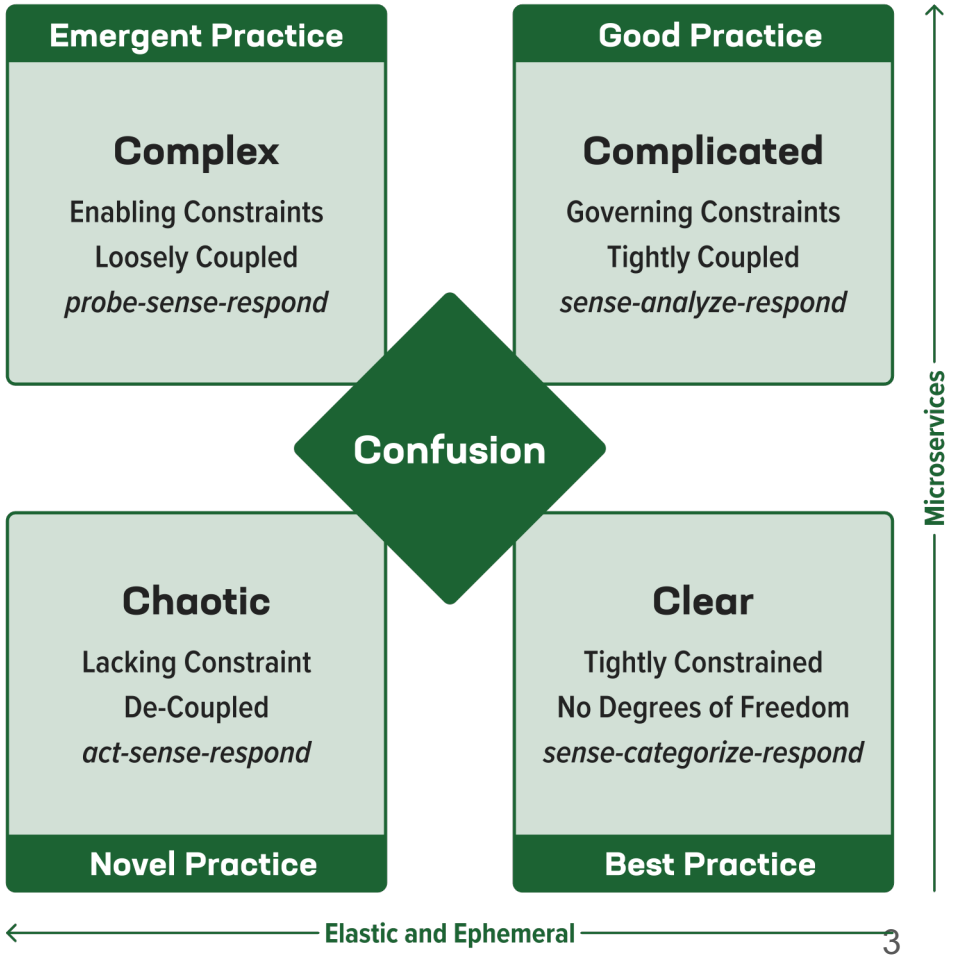


# But They Add Challenges

Especially when we consider this in a cloud

- Microservices create complex interactions.
- Failures don't exactly repeat.
- Debugging multitenancy is painful.
- So much data!




Cynefin Framework

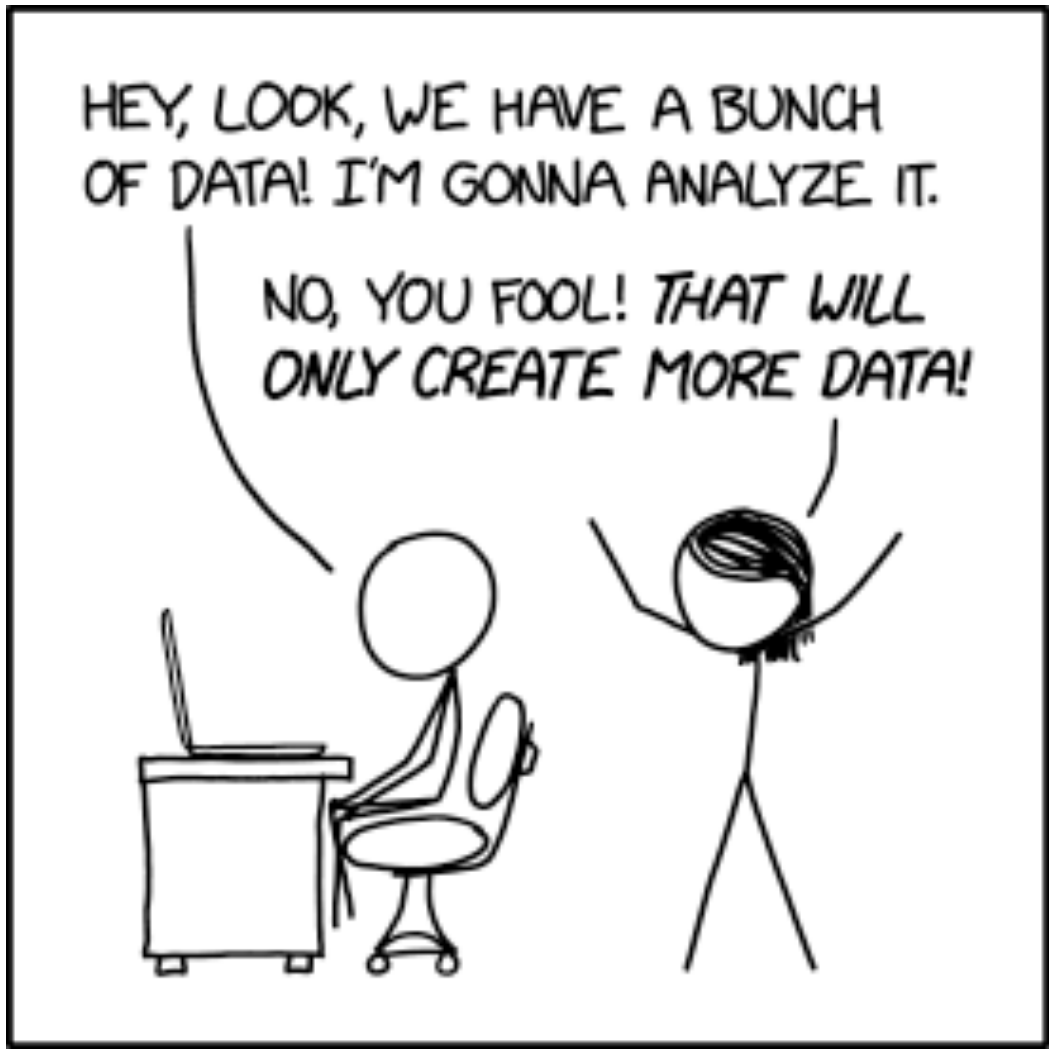
# Observability is a data problem

- AI/ML-driven Directed Troubleshooting
- Unlimited Cardinality
- Streaming data (near real-time)
- Open standards, open source data ingest
- Noise

*The more observable a system, the quicker we can understand why it's acting up and fix it*

# The problem data - there's so much of it



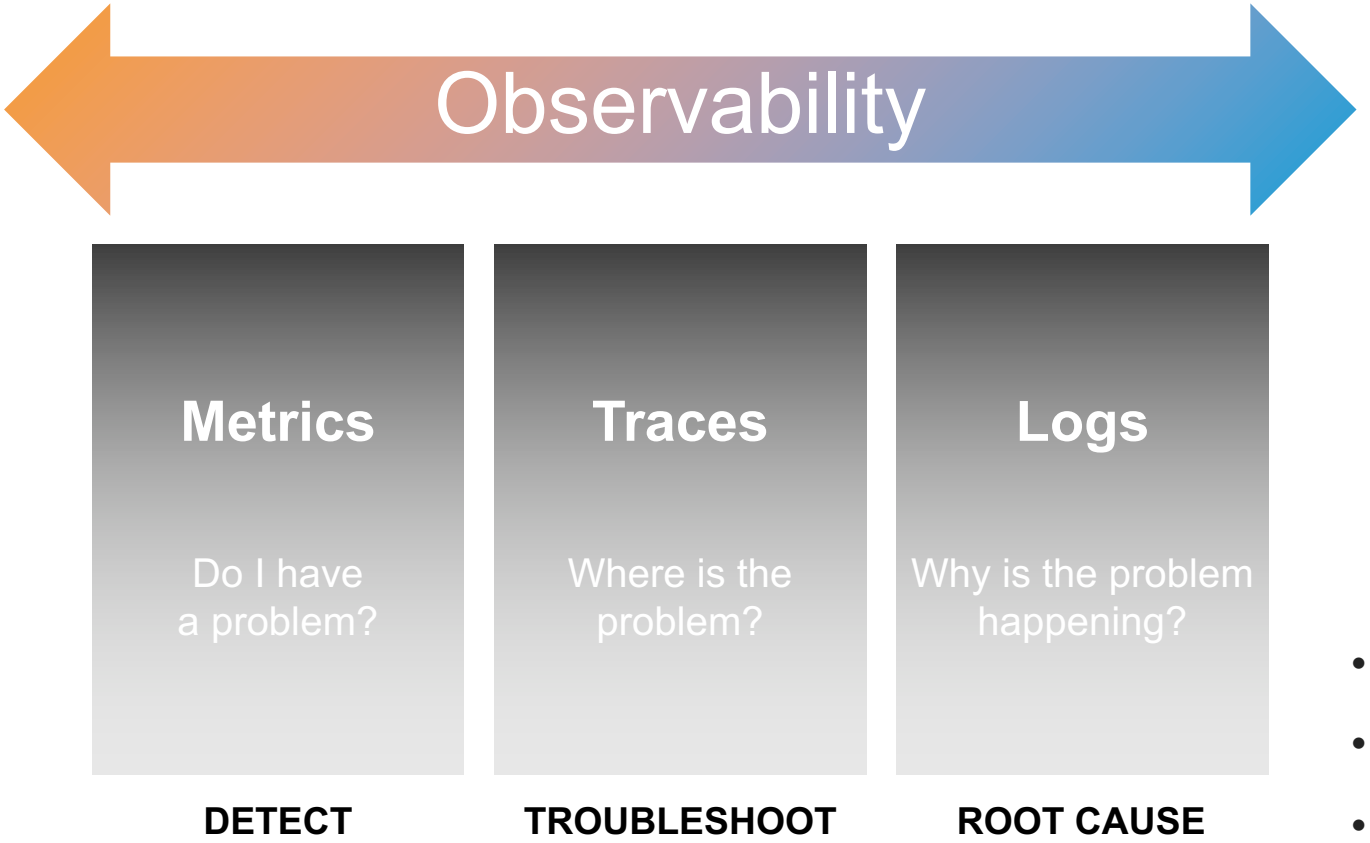


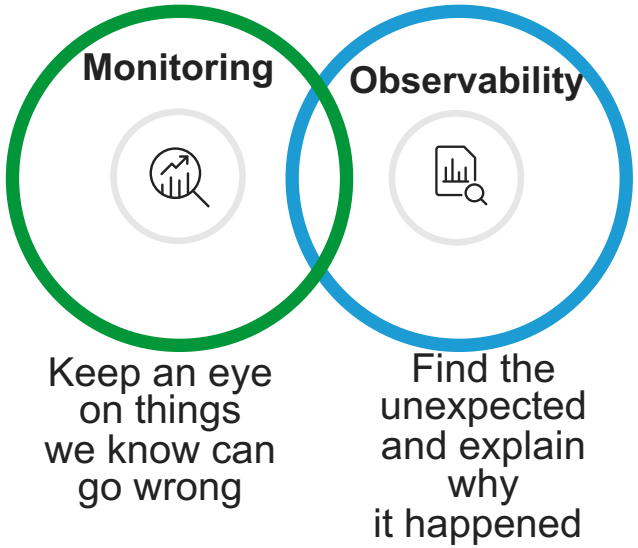

xkcd: Data Trap

# Observability -

Observability helps detect, investigate and resolve the *unknown unknowns* – FAST




- Better visibility to the state of the system
- Precise and predictive alerting
- Reduces Mean Time to Clue (MTTC) and Mean Time to Resolution (MTTR)

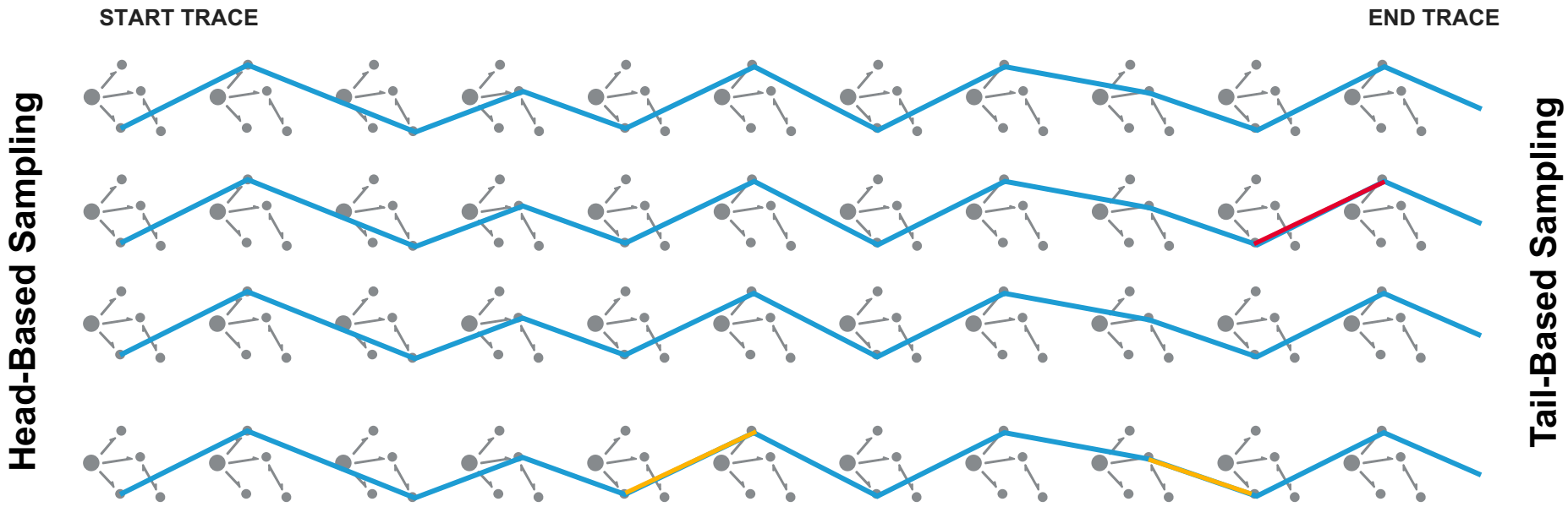
# Rule 1:
# You need to have all the data you need, even before you know you need it

# Dealing with the noise

- Filter the Signals
  - Linear, Low-pass, Band-pass, All-pass
- Sample the signals
  - Random, Head-based, Tail-based, Post-predictive, Dimensionality reduction
- Improve the visualization
  - Smart aggregation

# Tracing examined



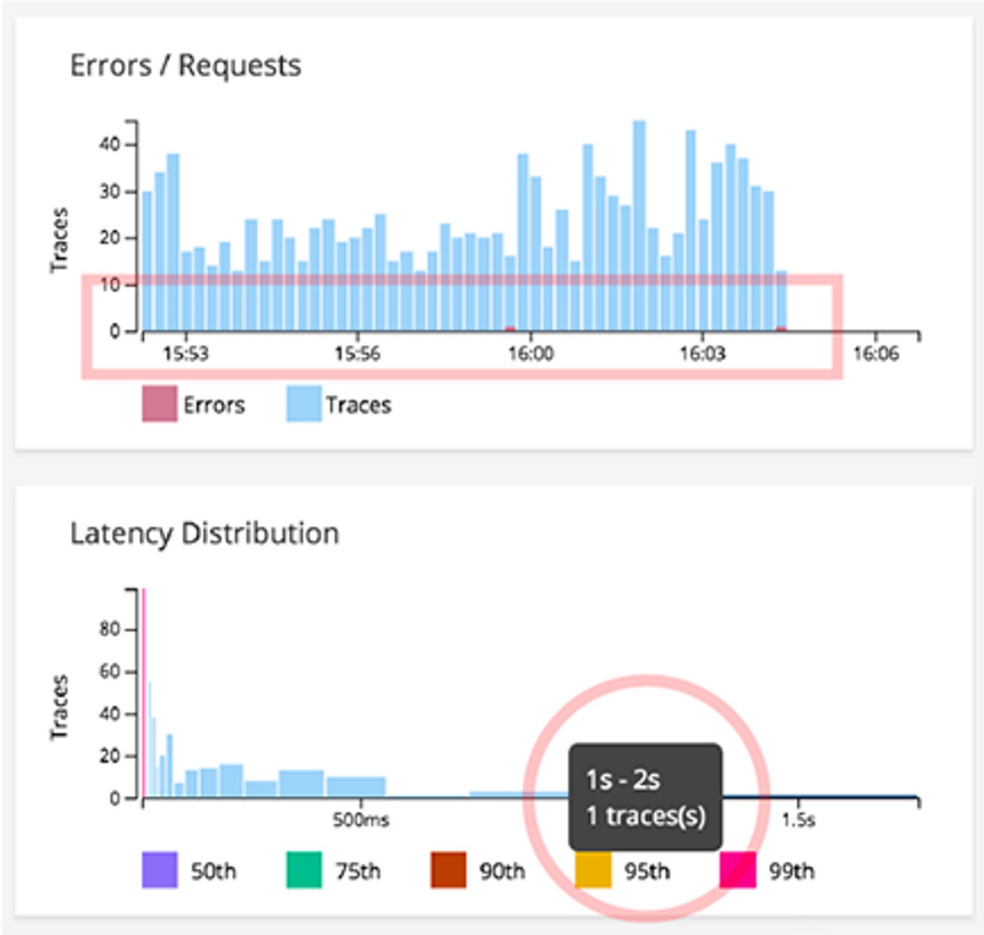

Purely random

Retention rate = Coverage

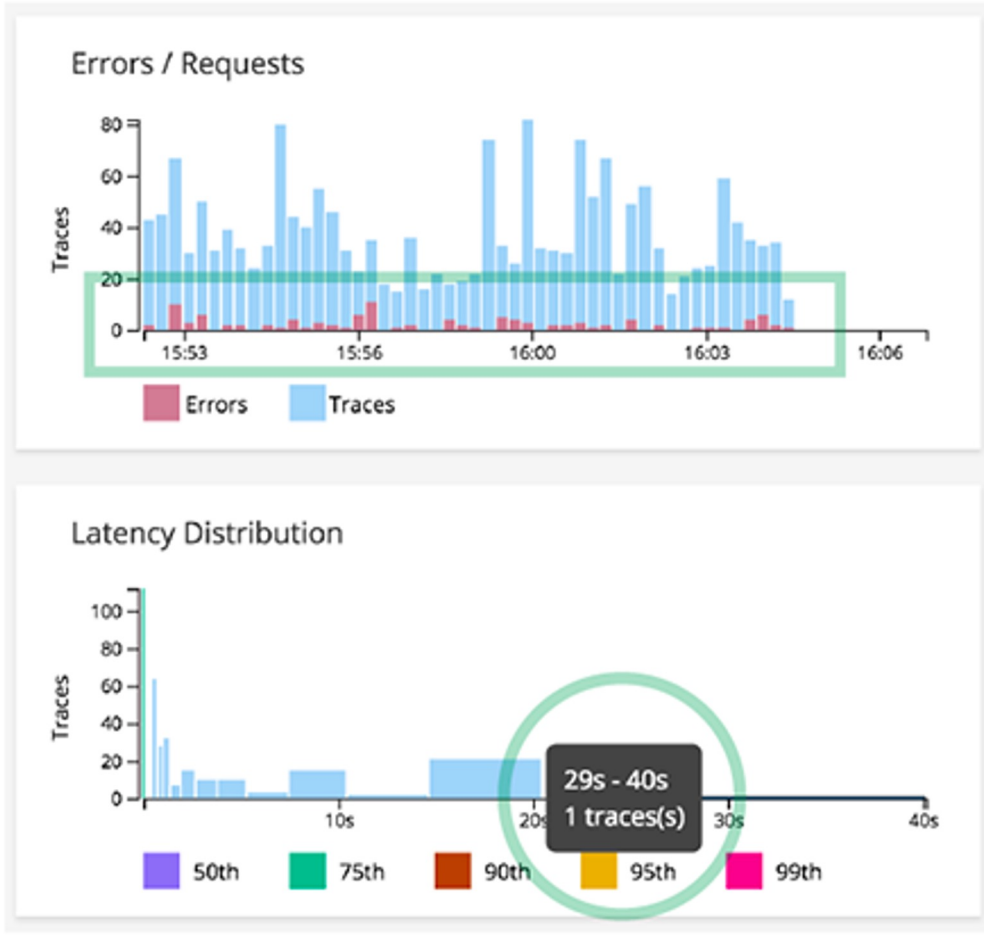
Not Random

Fault rate = Coverage

# Sampling

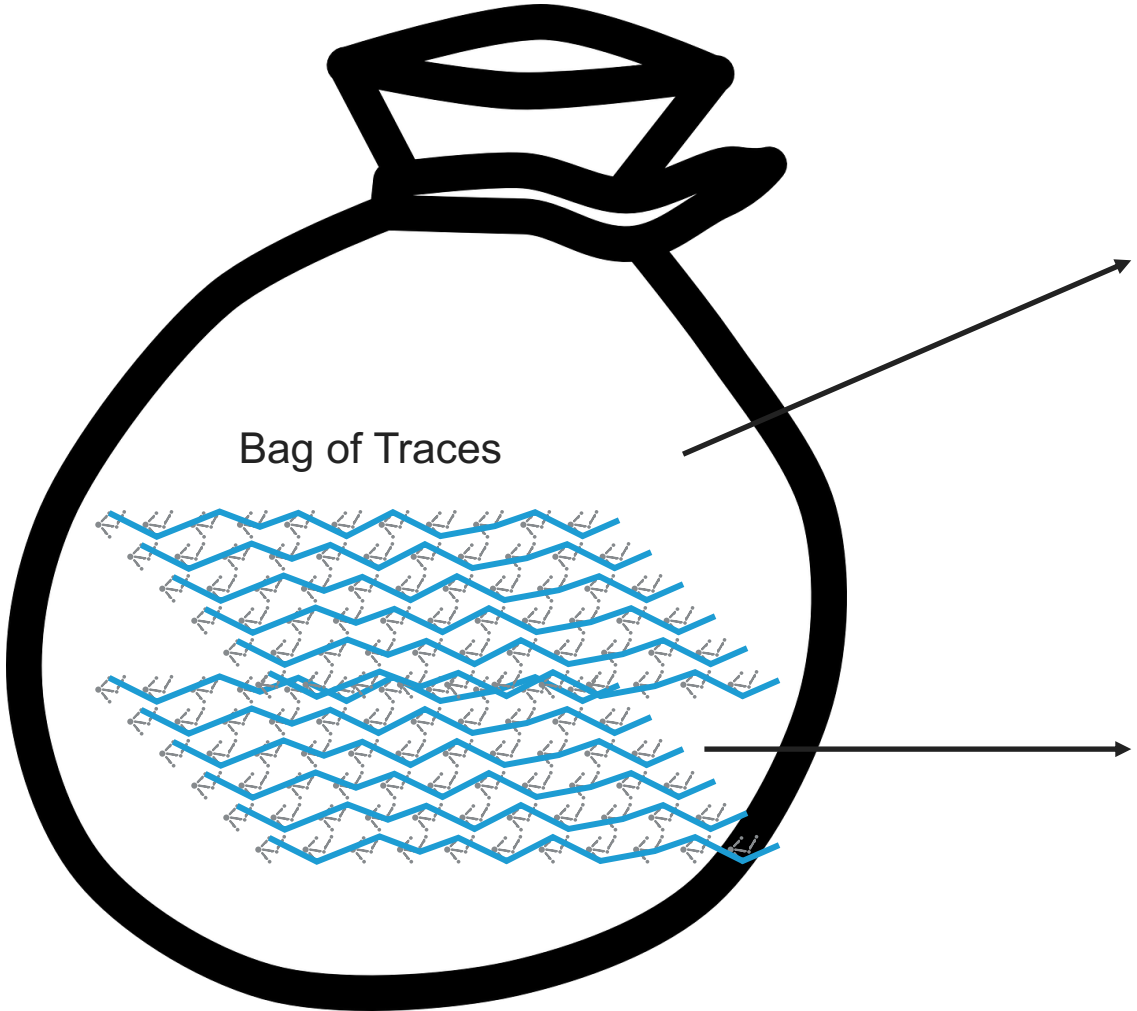


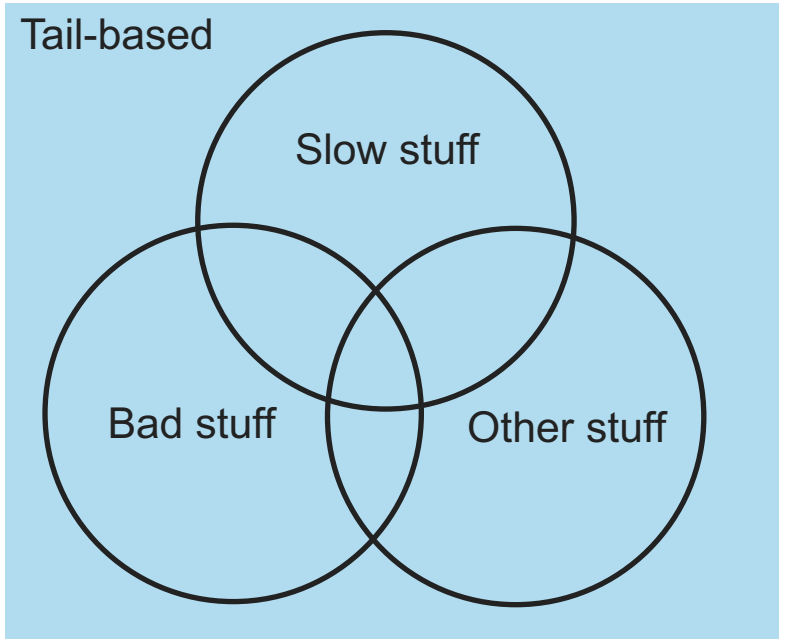

# No Sampling



# A couple of approaches: Just grab the needed stuff



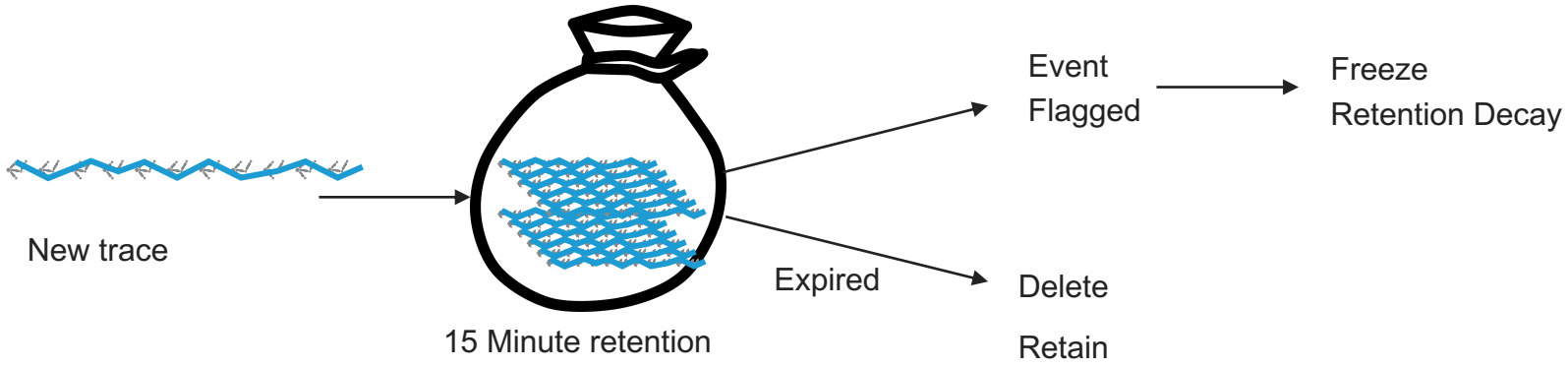





# A couple of approaches: Post-predictive

- Post-predictive
  - Keep everything for a relatively short period of time.
    - Usually, a rolling period
    - This is tricky: Debug, Security, Customer Status
  - If an event of interest occurs, retain all existing data
  - Much like log rotation, you can decide to archive off in chunks.
    - Dimensionality reduction could play a role here



# But wait! My metrics tell me everything

Your metrics are usually not sampled, for your infrastructure

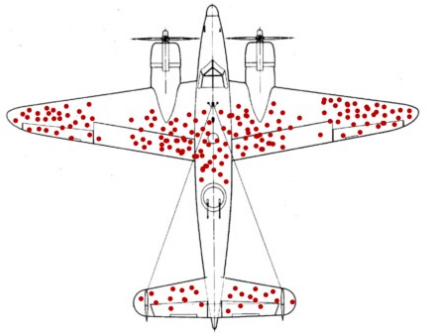
But can be for your application traces

Leading to bad duration results and potential missed alerts

# Data is your Observability Partner

- Your ability to use observability is dependent on your data
- Don’t let the “chosen data” bias your results
- Keep the right stuff
  (This might be all the stuff)



By McGeddon - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=53081927
(Image:
https://drive.google.com/file/d/1ESDKGDGHNjfBaQT4fOztR2sEnzaj1YUl/view?usp=sharing )

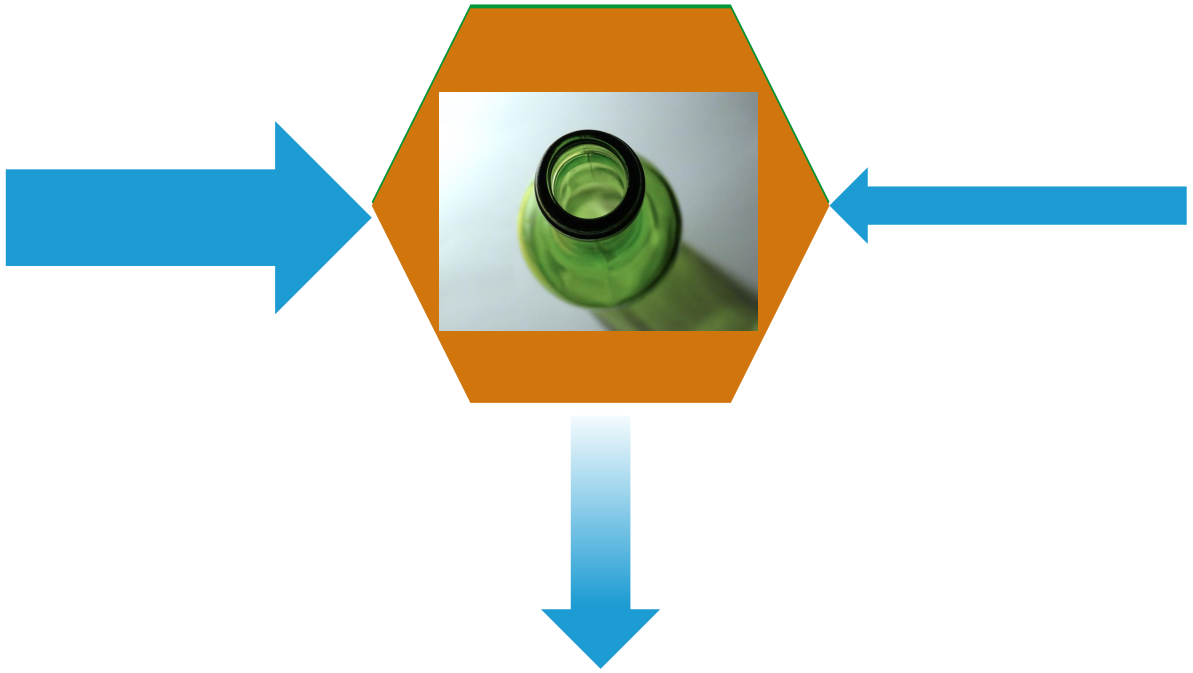
 

# Rule 2:
# The resolution and speed of the data directly impact the insights you gain

# Observability?
# Is your data lying?

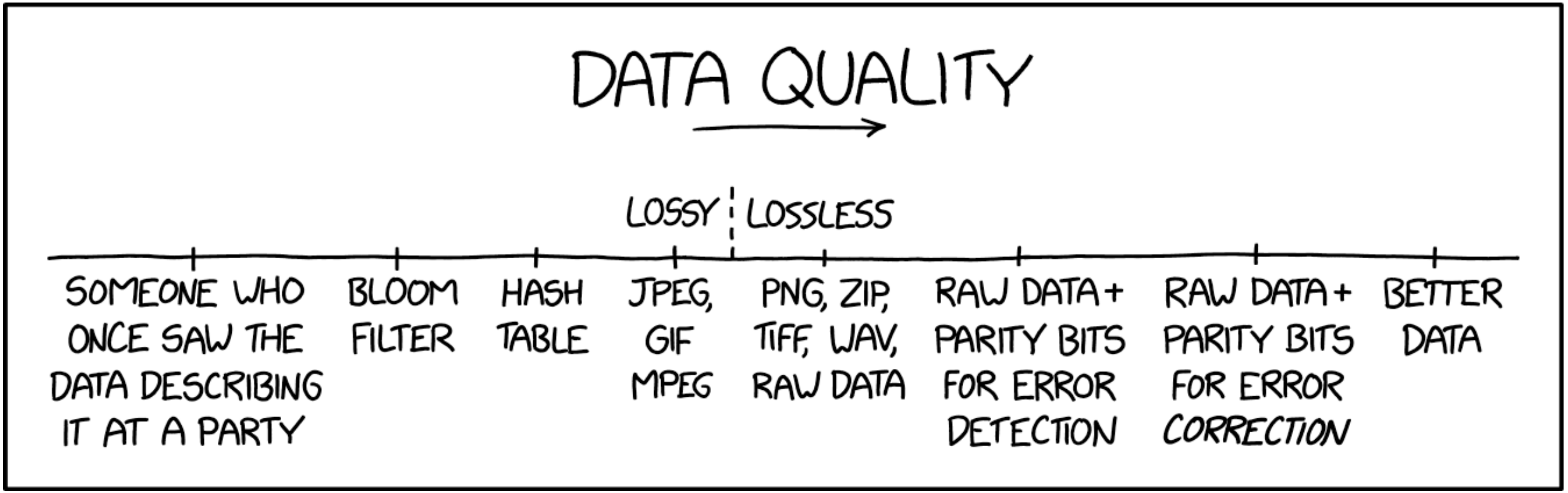
Resolution of Data
(Precision, Accuracy)



Speed of Data
(Deterministic Response)

Aggregation, Analysis
and Visualization of Data

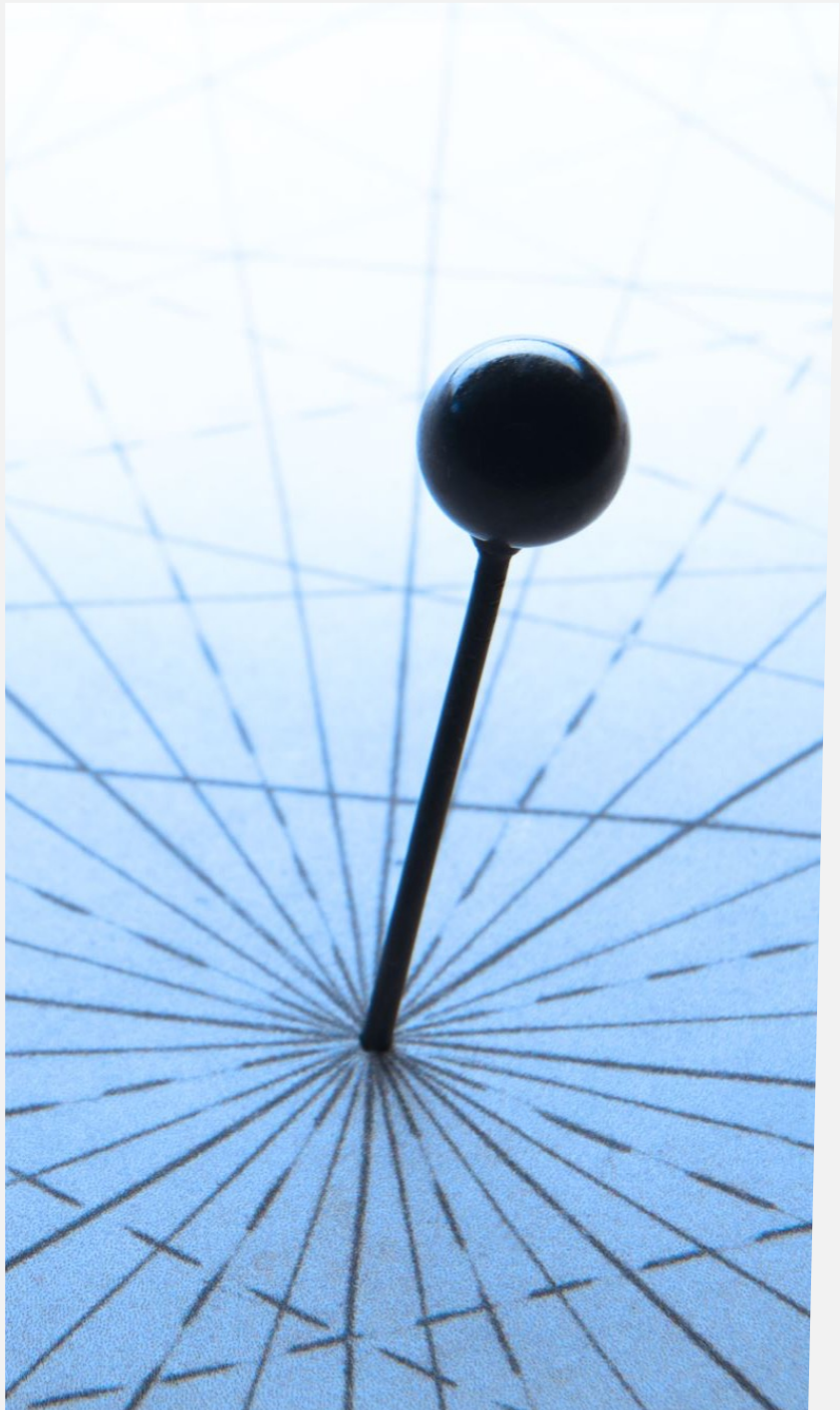
# The problem with data - there's so much of it






https://xkcd.com/2739



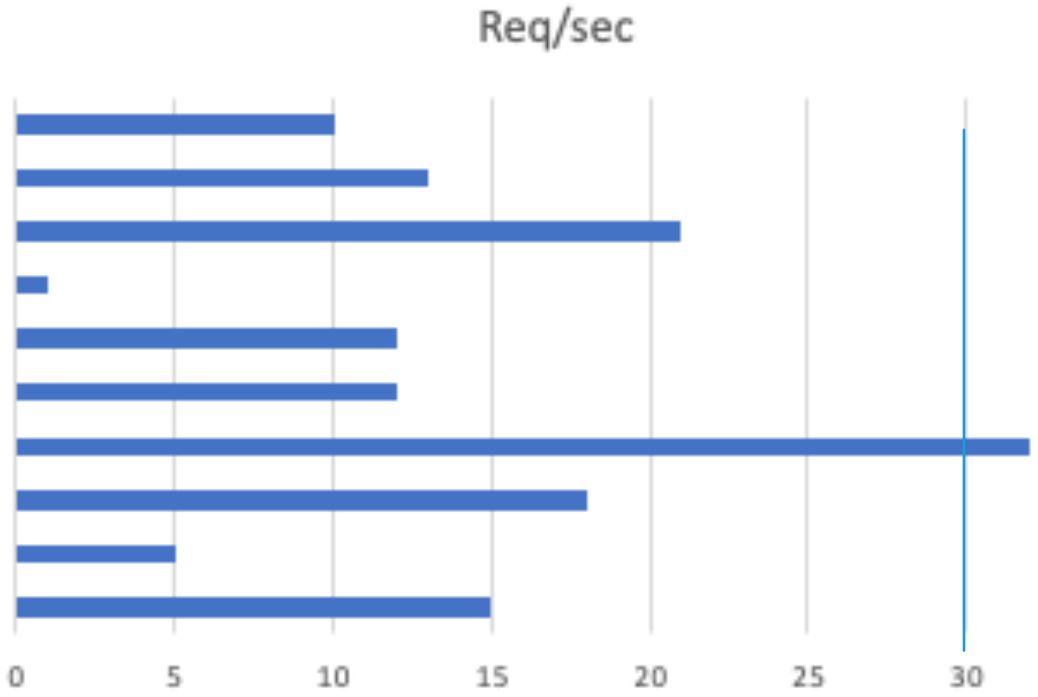
# Discussing accuracy and precision

- Interchangeable?
- Accuracy is that the measure is correct
- Precise means it is consistent with other measurements

Observability depends on both

**But aggregation and analysis can skew this**

# And sometimes aggregation is not your friend
# Missing the point




**10 sec average =13.9**

**95% = 27.05**

**First 5 sec average =16.4**

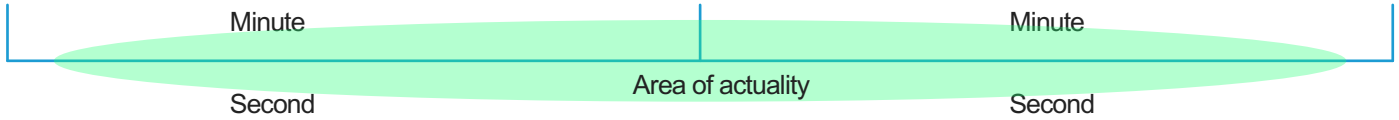
**95% = 29.2**

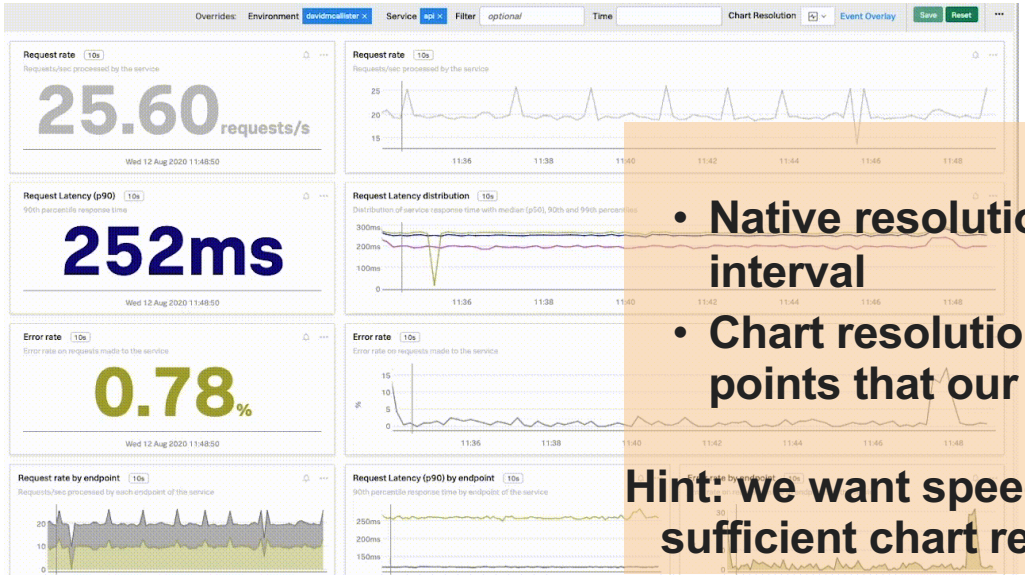
**Second 5 sec average =11.4**

**95% = 19.4**

# Data resolution, Reporting resolution
# Chart resolution, Native resolution

- Always deliver all data points regardless of reporting
- Finer granularity means more potential precision



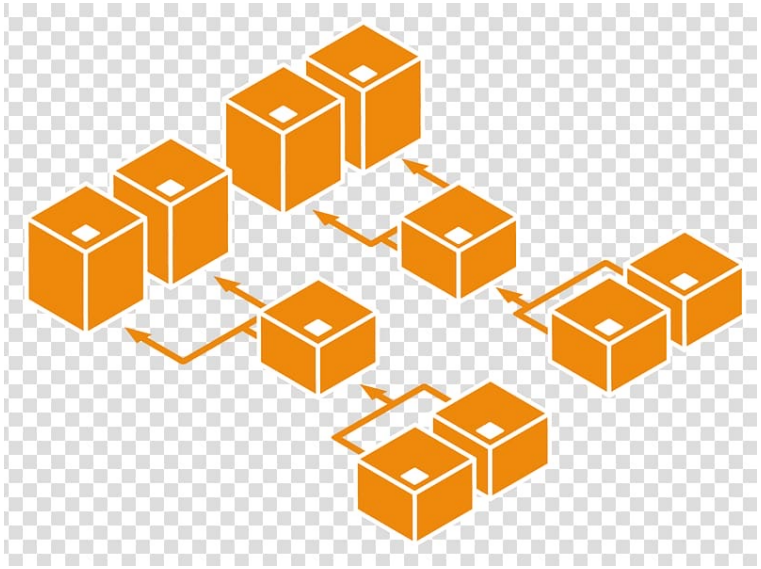




- **Native resolution is our data collection interval**
- **Chart resolution is the aggregation points that our graphs use**

**Hint: we want speedy data collection and sufficient chart resolution**

# Add in Complexity

- Cloud-compute Elasticity



- Ephemeral Behavior

Drift and Skew

# Drift/Skew? Accurate Timestamps

- Network latencies get lower
- Event frequencies are higher
- Chrony on AWS/GCP
  - ~10s to 100s of accuracy
  - May not always order events properly

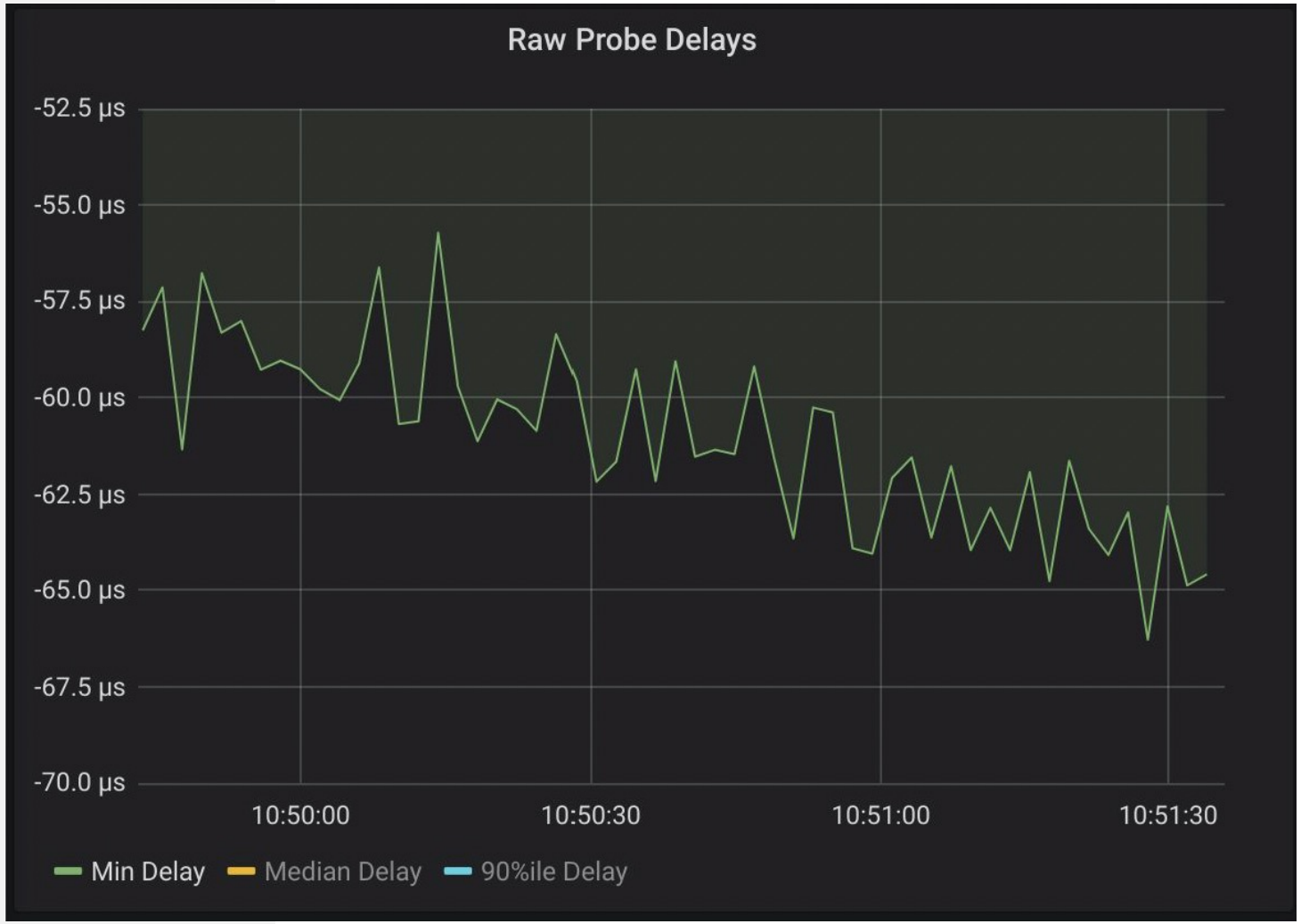


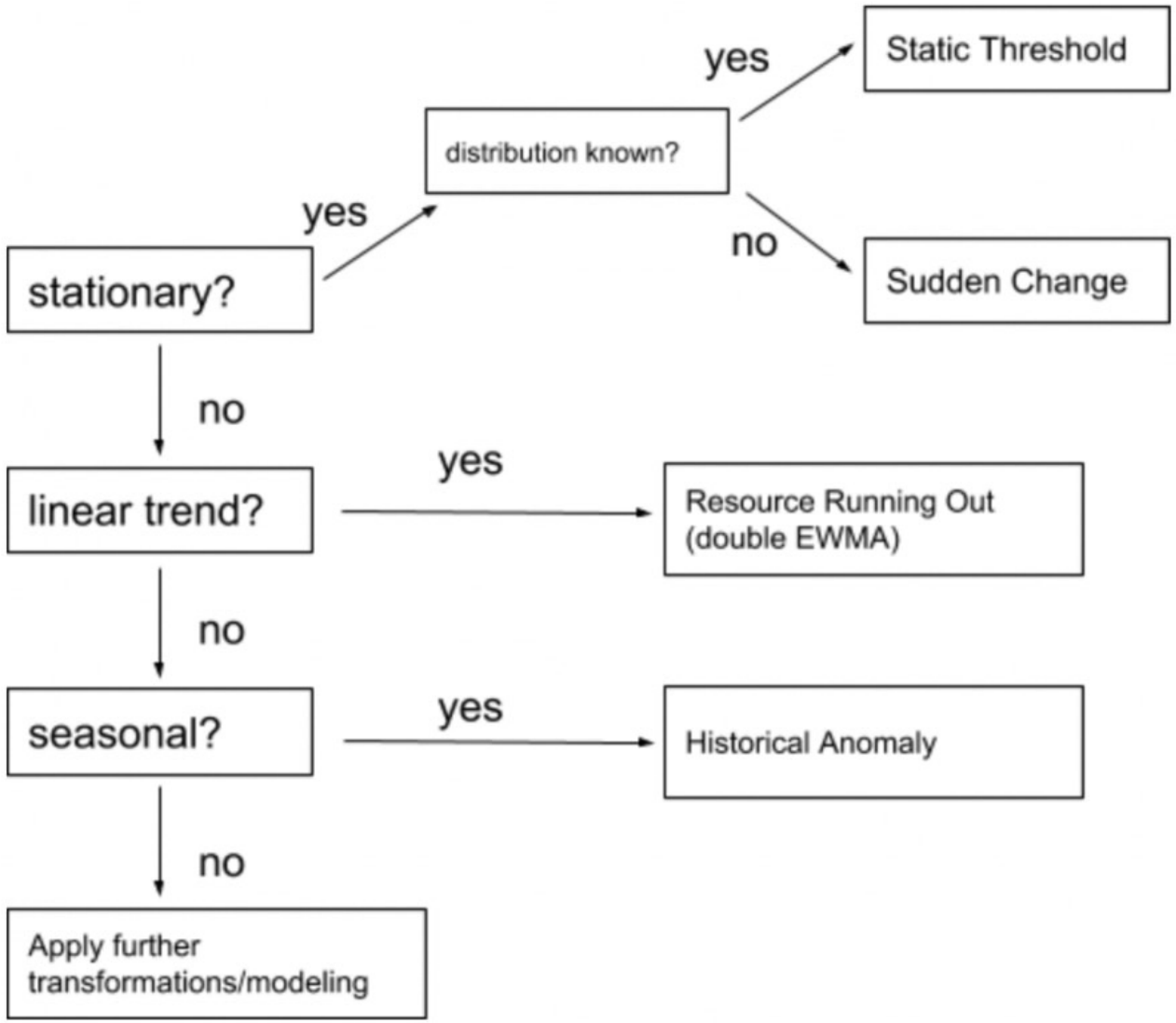

Image: ClockWork.io

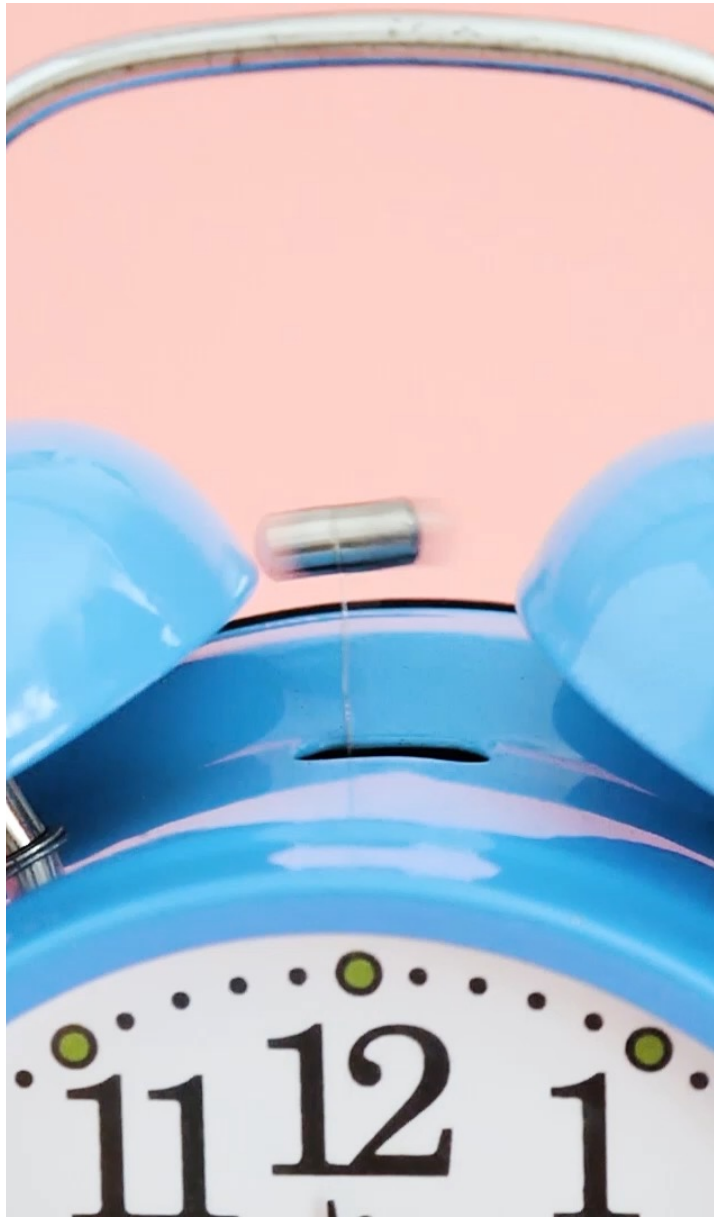
# Aligned traces?

- Spans may start ahead of parent spans starts
- Spans may start after parent span ends
- Span durations can be impacted, resulting in lack of precision


Image: ClockWork.io

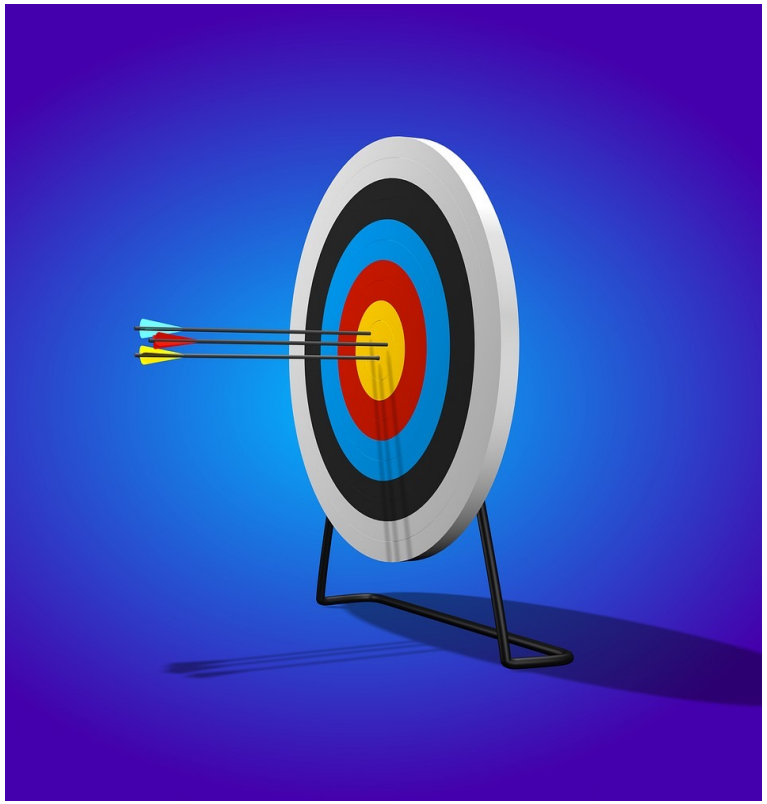
# Predictive and response alerting



# Predictive behavior

- Prediction is only as good as the data precision and accuracy
- Historic versus Sudden Change
- (Trend) Stationary
- Expect false positives (and negatives)

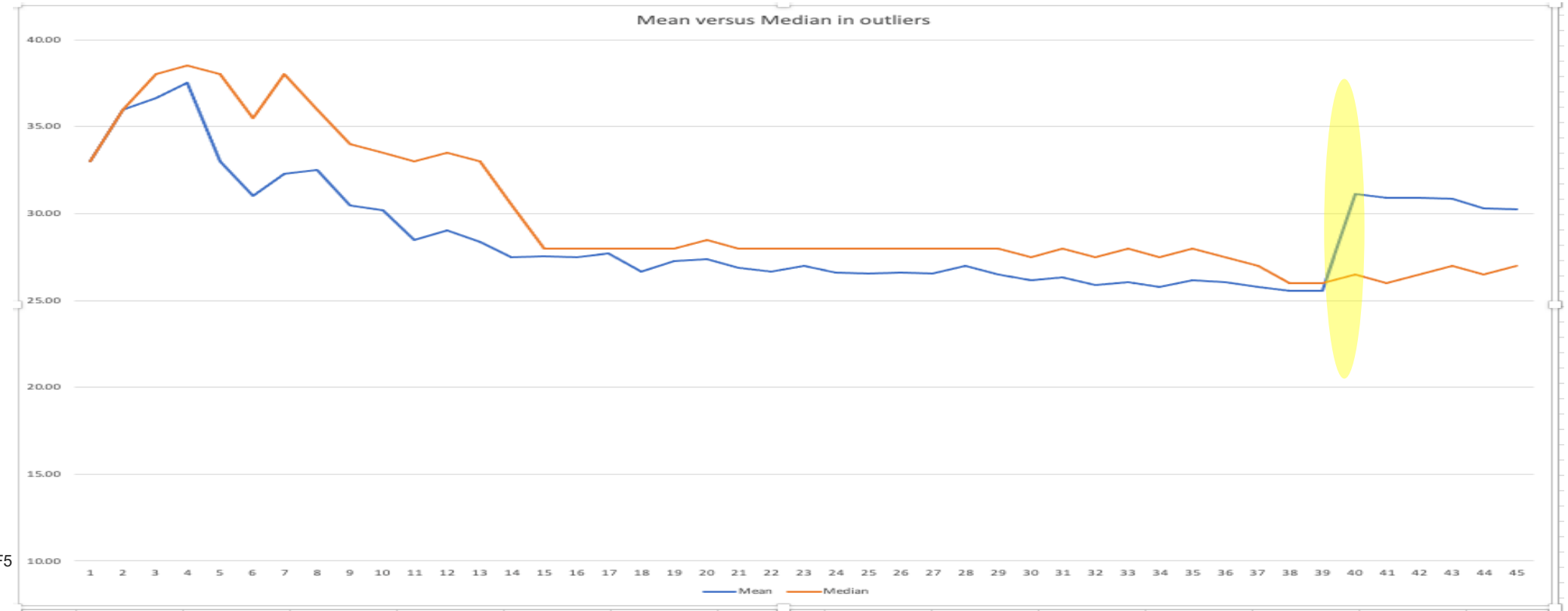


# Choosing Mean and Median

This matters when looking for the unexpected

- For sudden change, mean is likely a better choice
- For historic anomaly, median might be better
  - Dependent on your timescale

| Old | | New |
|---|---|---|
| | Outlier | 250 |
| 25.54 | Mean | 31.15 |
| 26 | Median | 26.5 |

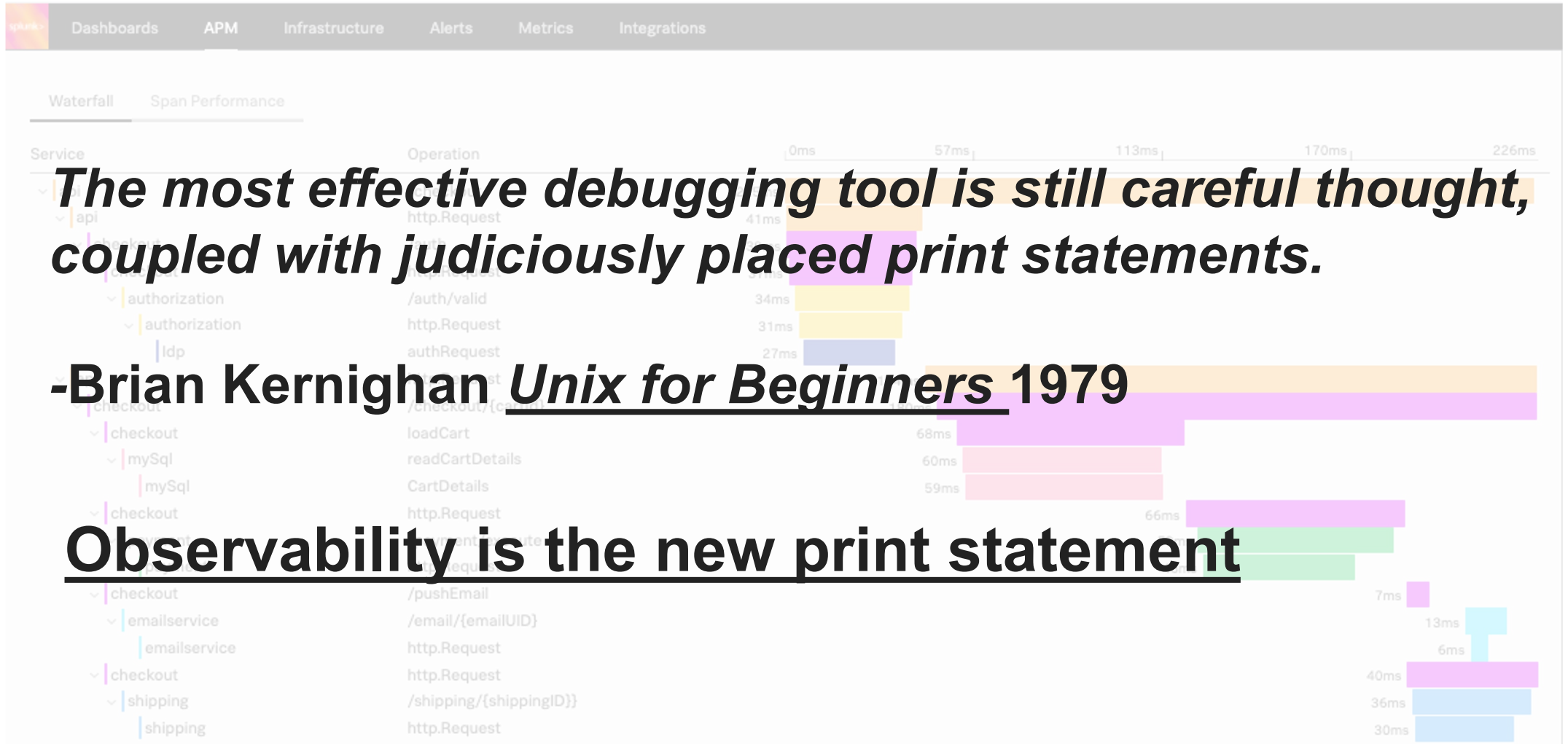


# Speed and resolution review

- Observability is only as useful as your data's precision and accuracy
- Your consideration of the data needs to account for elastic, ephemeral and skew
- Prediction is a target
- But not the only one (MTTC, MTTR, customer happiness)

# Closing Thoughts




***The most effective debugging tool is still careful thought, coupled with judiciously placed print statements.***

**-Brian Kernighan *Unix for Beginners* 1979**

**Observability is the new print statement**

# Thanks for listening

- https://www.linkedin.com/in/davemc