Techniques and Tools for

A Coherent Discussion About Performance

in Complex Systems

Performance Must Matter

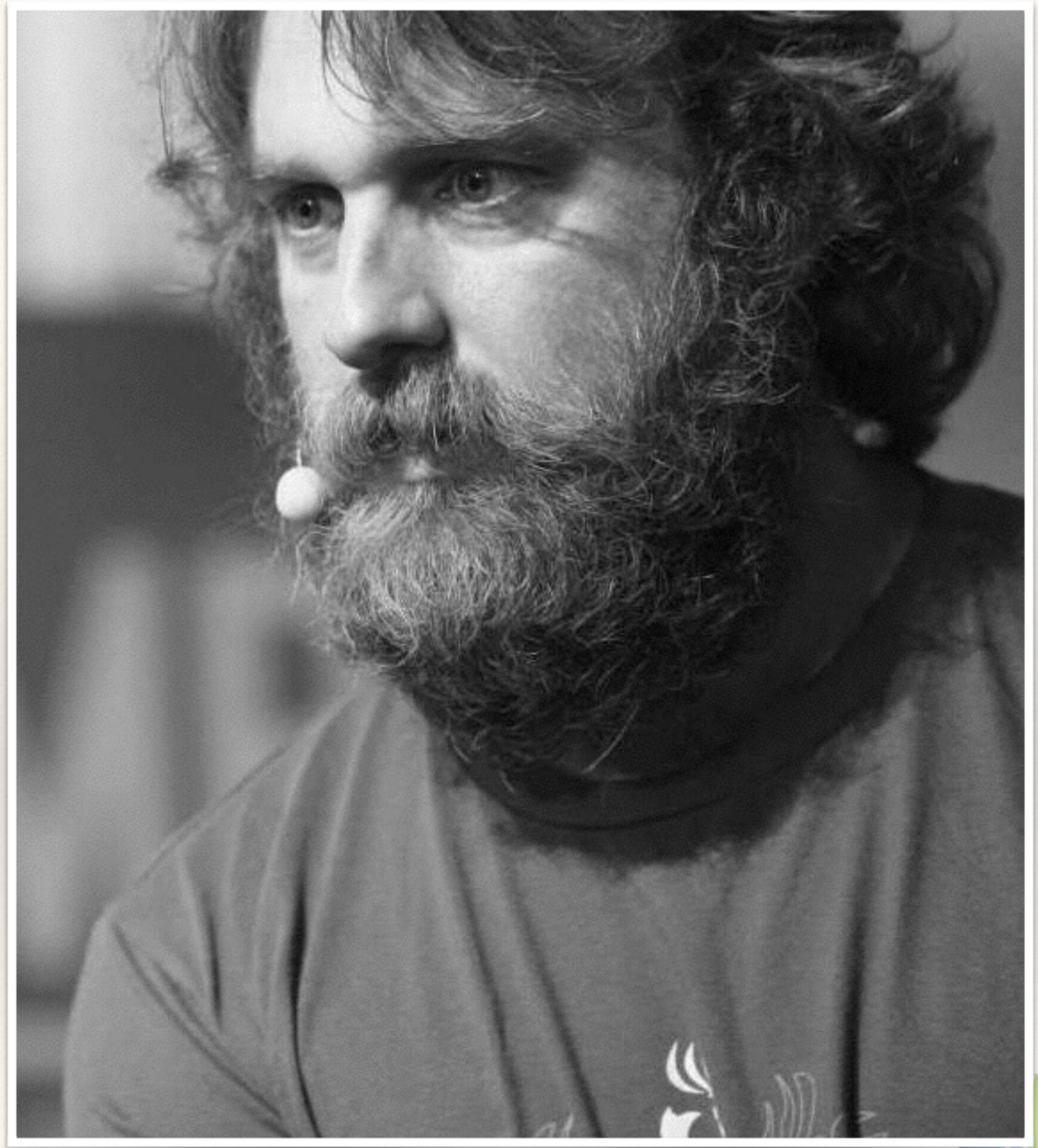
First it must be made relevant.

Then it must be made important.

If you don't care about Performance

You are in the wrong talk.

@postwait should throw you out.



Perhaps some justification is warranted

Performance…

makes a better user experience
increases loyalty
reduces product abandonment
increases speed of product development
lowers total cost of ownership
builds more cohesive teams

Consistent Terminology

Inconsistent terminology is the best way to argue about agreeing

RFC: http://l42.org/GwE

Define: Monitoring

Discusses:

components, systems, observability, agents, static and dynamic properties

“Monitoring is the action of observing and checking static and dynamic properties of a system.”

–Heinrich Hartmann

tl;dr it's all about latency…

Throughput vs. Latency

Lower latency often
affords increased throughput.

Throughput is a well tread topic
and uninteresting.

Latency is the focus.

“Latency is the mind killer.”

–Artur Bergman

Generally, time should be measured in seconds.
UX latency should be in milliseconds.

Time

Users can't observe microseconds.

Users quit over seconds.

Users experience is measured in milliseconds.

That said: seconds are the clearest international unit of measurement. Use non-integral seconds.

“Time is an illusion.
Lunchtime doubly so.”

–Douglas Adams

“Seconds are the clearest unit of time measurement.
Use non-integral seconds for measuring time.
Convert for people later.”

–Theo Schlossnagle

Music is all about the space between the notes.

Connectedness

Performance is about how quickly you can complete some work.

In a connected service architecture, performance is also about the time spent between the service layers.

Developing a

Performance Culture

It is easy to develop a rather unhealthy performance culture.

Focus on

Small Individual Wins

Report on and celebrate

Large Collective Wins

What's next?

The Future of Systems Observability

Have a deeply technical cross-team conversation about performance

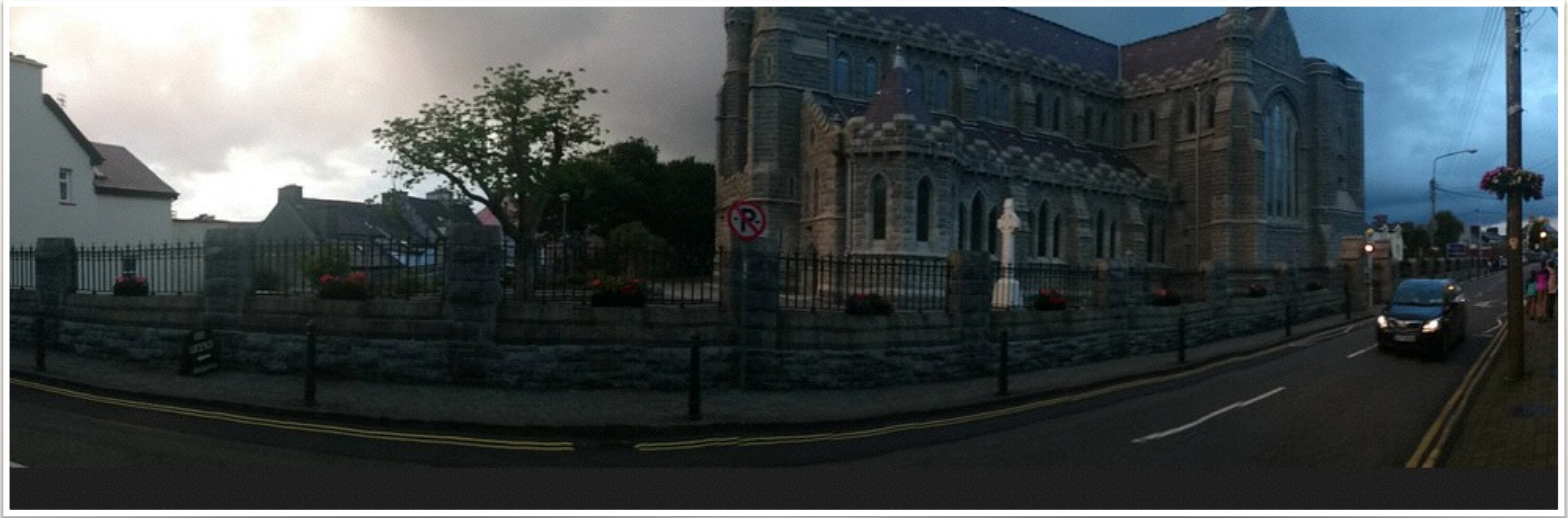
To predict the future, we look to the past.

Web monitoring:

- [2000]-> Synthetic Monitoring
- [2010] -> RUM

Systems monitoring:

- [2010] -> Synthetic Monitoring
- [????] -> Observed Behavior Monitoring



A search for the best representation of behavior

To win, we must compromise

To conquer our information-theoretic issue, we must take a different approach.

Path 1

Full system tracing. Sometimes.

Fun…

The way for deep contextual truth.

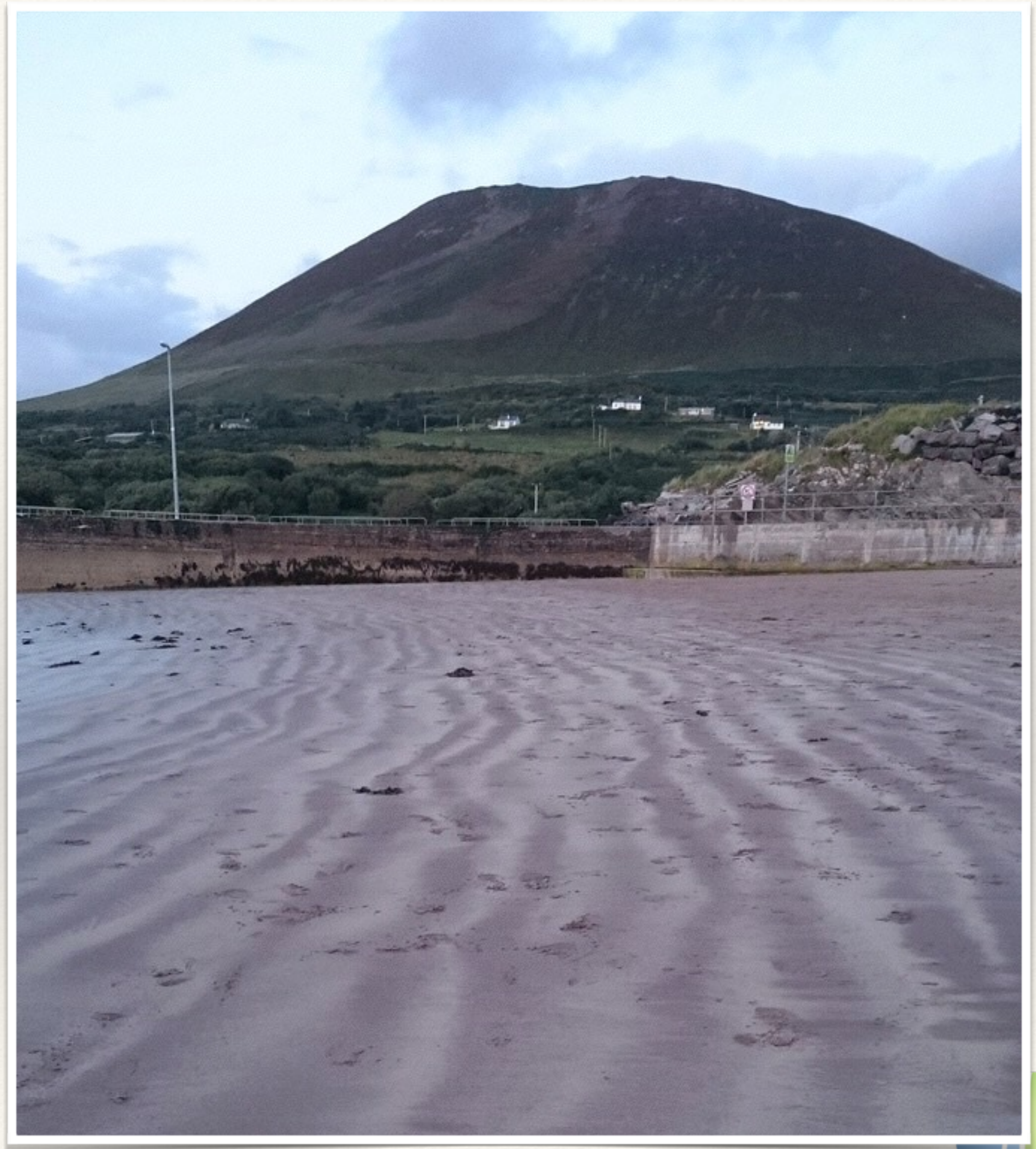
Often dirty and expensive.

Path 2

Keep the volume,
Lose the dimensionality.

You can't find where
each grain of sand came from.

But you can
understand an accurate topology
of the beach over time
and reason about it.



Path 1

Tooling must transcend the team

and keep conversations consistent

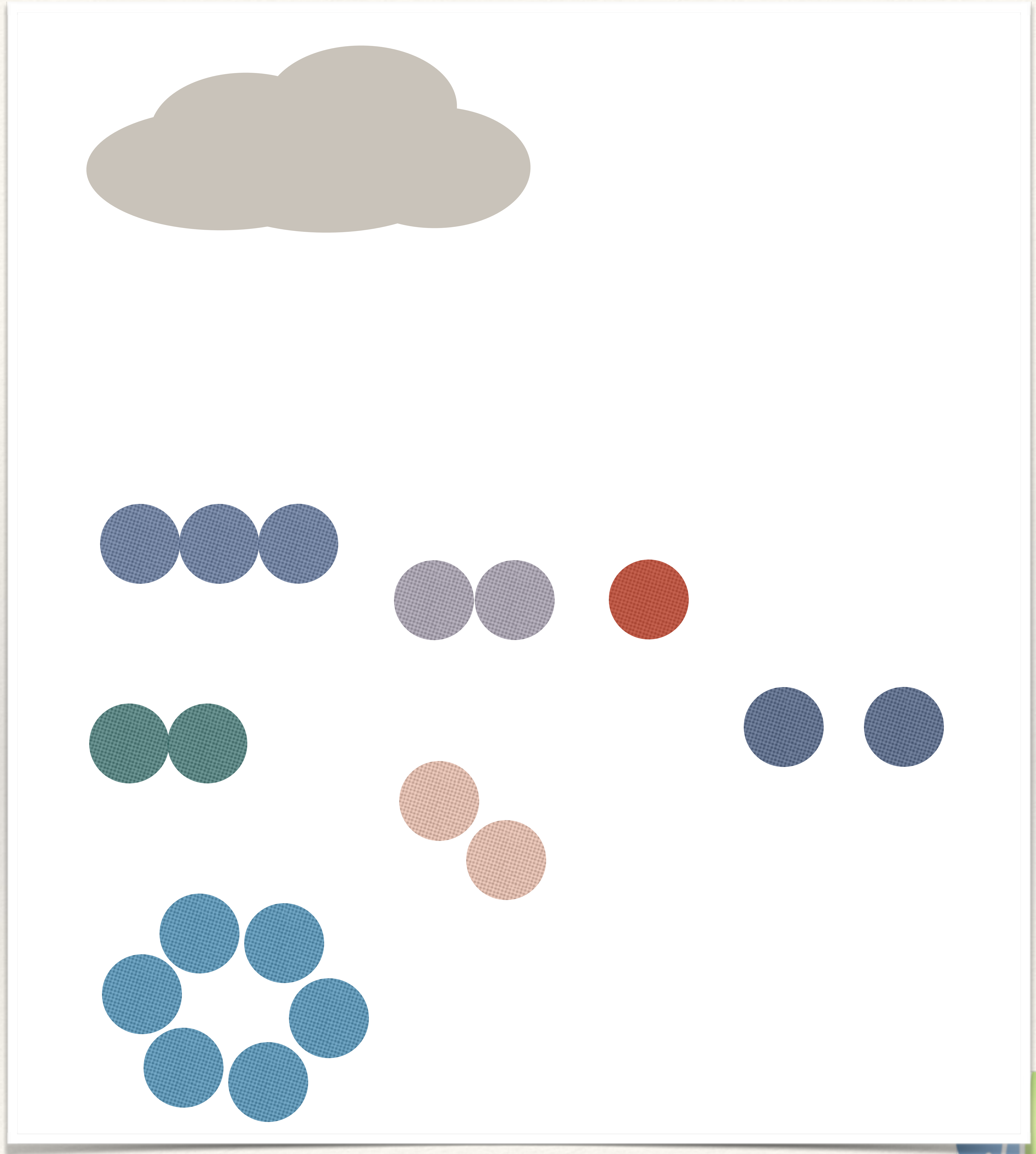
Large-Scale Distributed Systems Tracing Infrastructure

Dapper

Google published a paper:

research.google.com/pubs/pub36356.html

As usual, code never saw the outside.



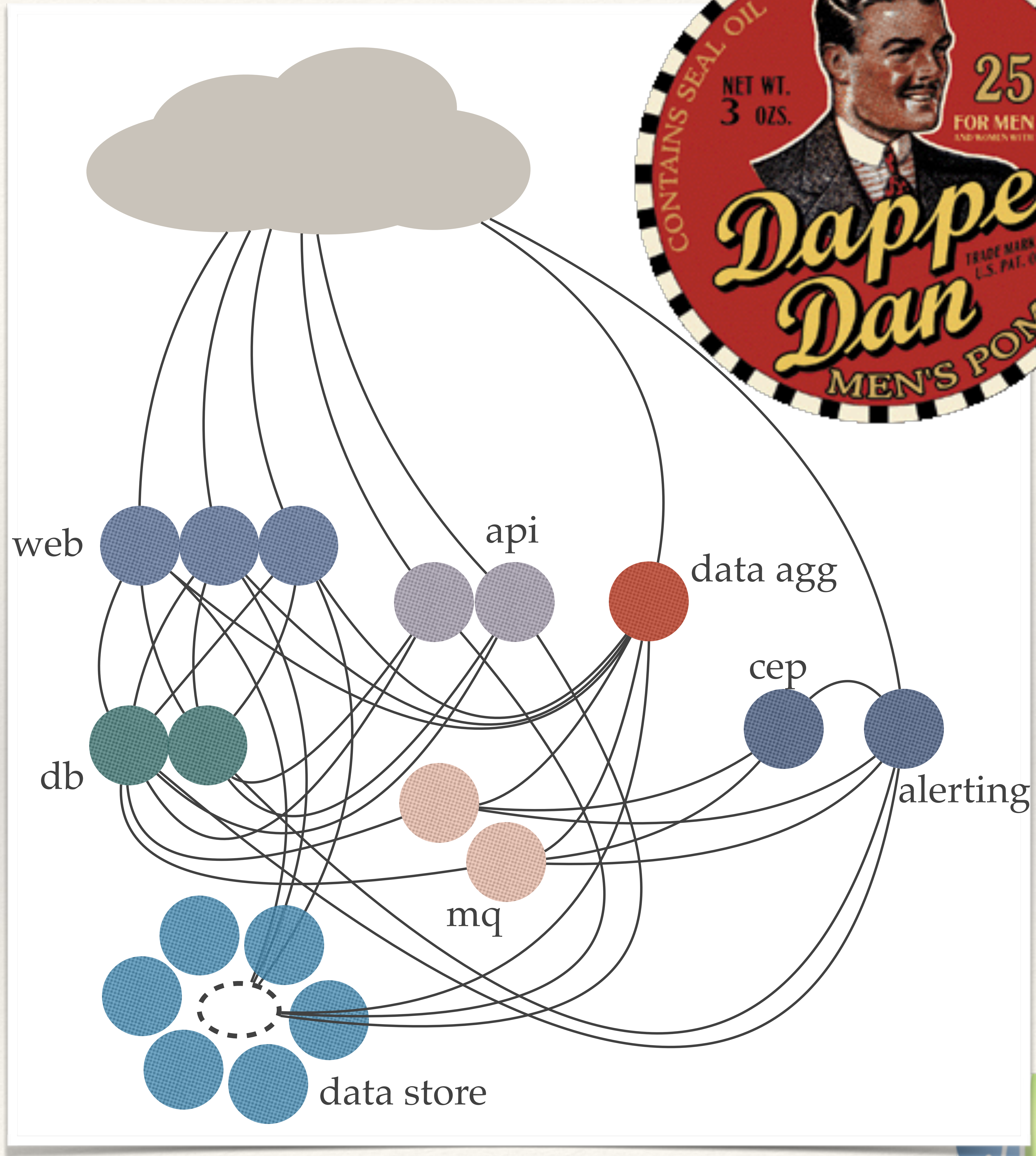
Large-Scale Distributed Systems Tracing Infrastructure

Dapper

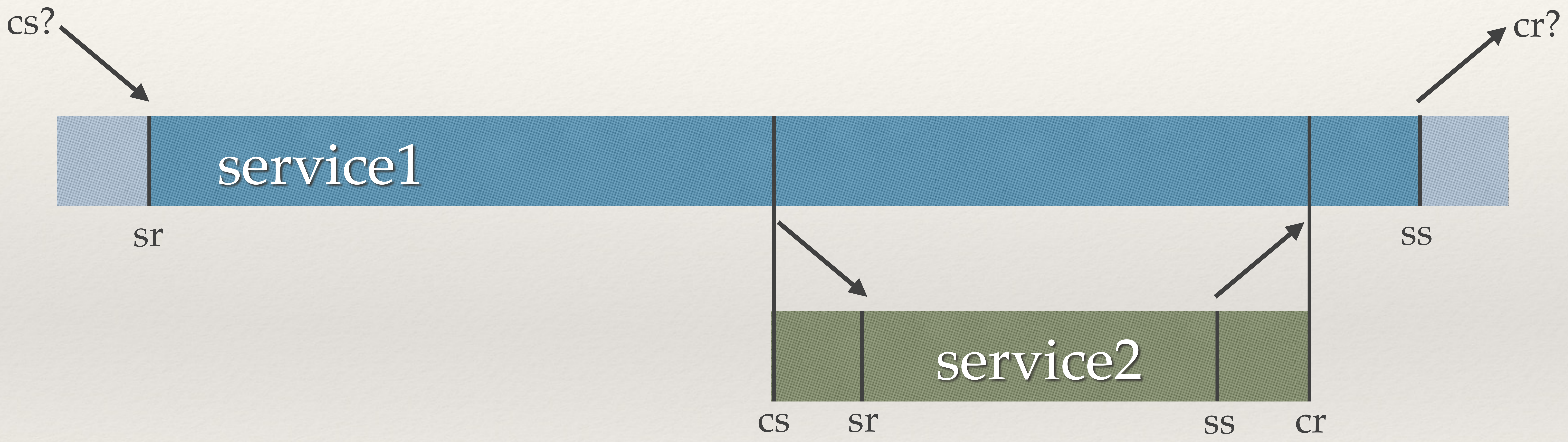
Google published a paper:

research.google.com/pubs/pub36356.html

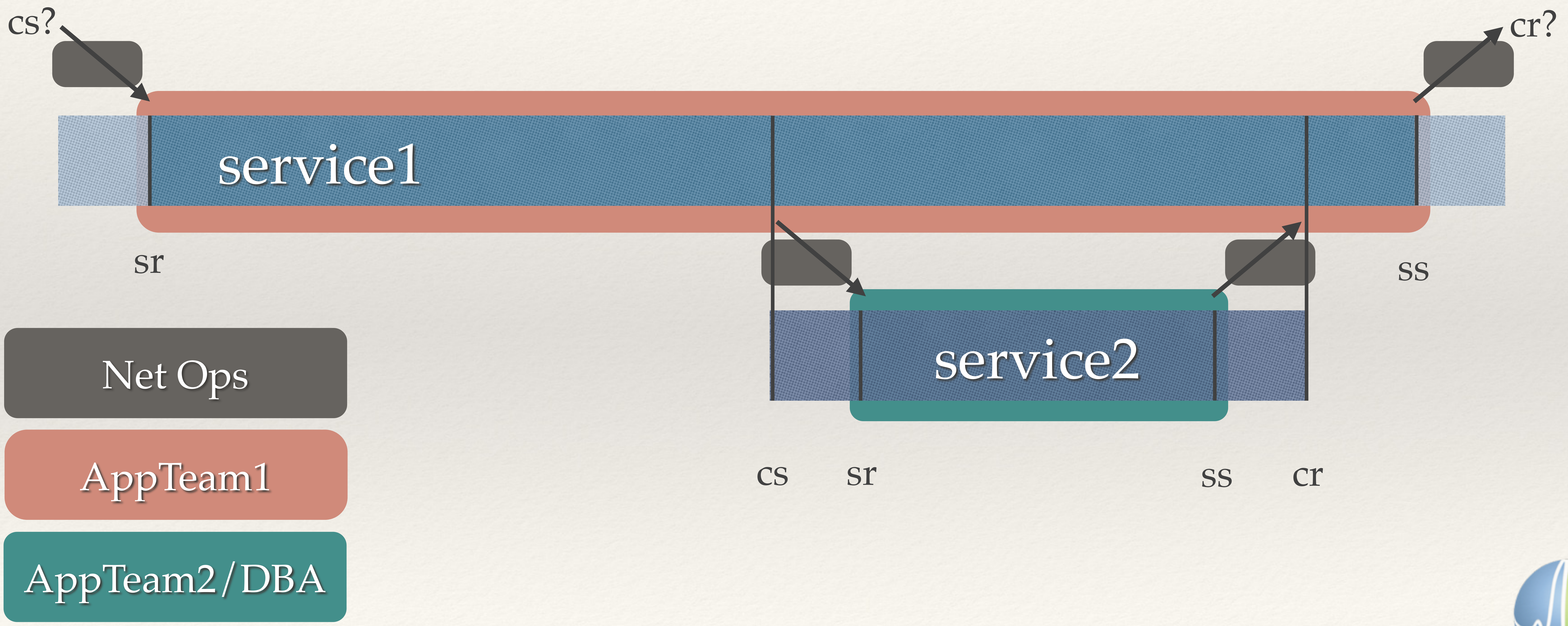
As usual, code never saw the outside.



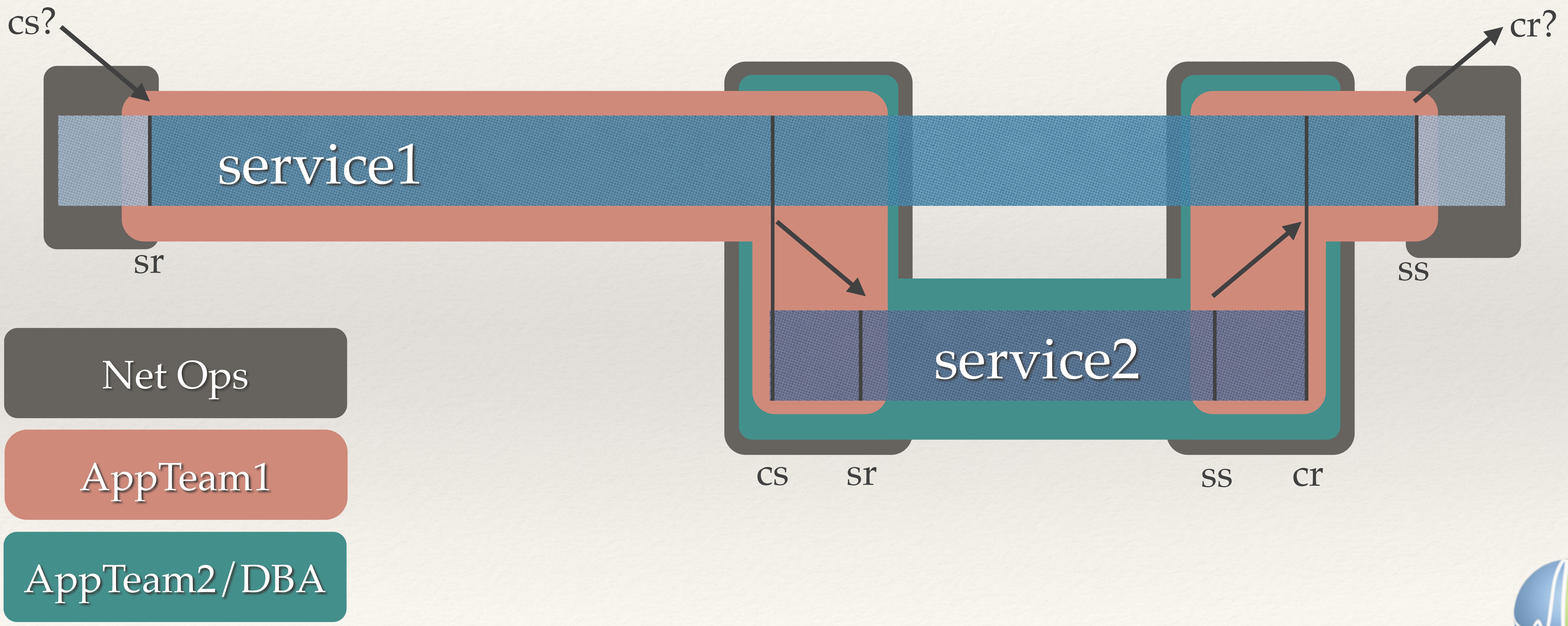
Visualization



Siloed Teams



Better Responsibilities



This doesn't work at all levels

Imagine Service "Disk"

If you trace into each disk request and record these spans…

we now have an information-theoretic issue

A pseudo-Dapper

~~Zipkin~~ OpenZipkin

Twitter sought to (re)implement Dapper.

Disappointingly few improvements.

Some unfortunate UX issues.

Sound. Simple. Valuable.

Thrift and Scribe should both die.

Scribe is Terrible

Terrible. Terrible Terrible.

Zipkin frames are thrift encoded.

Scribe is "strings" in Thrift.

Zipkin is Thift, in base64, in Thrift. WTF?

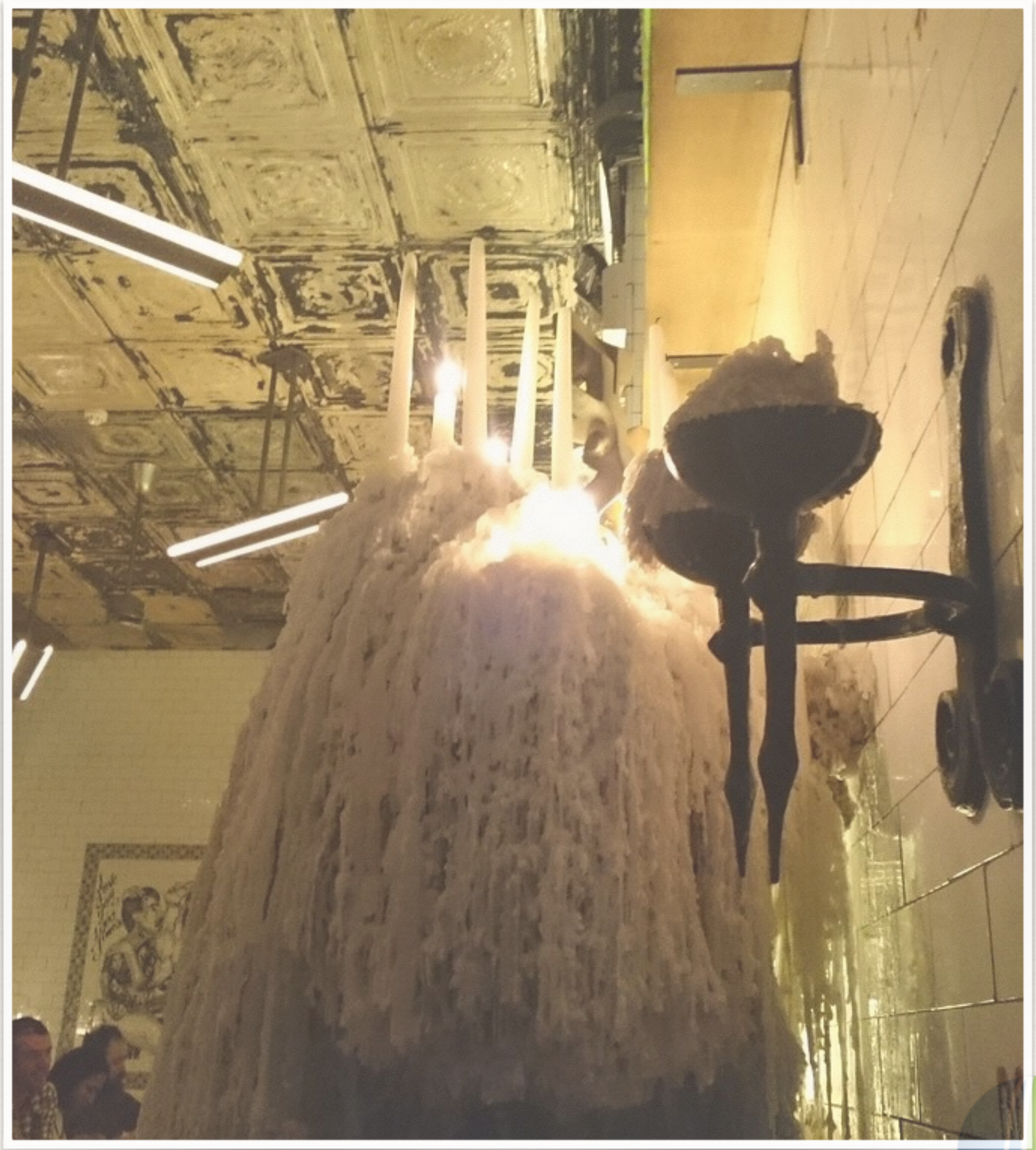
The whole point is to be low overhead

Screw Scribe

We push raw thrift over Fq
github.com/circonus-labs/fq

Completely async publishing,
lock free if using the C library.

Consolidating Zipkin's bad decisions:
github.com/circonus-labs/fq2scribe

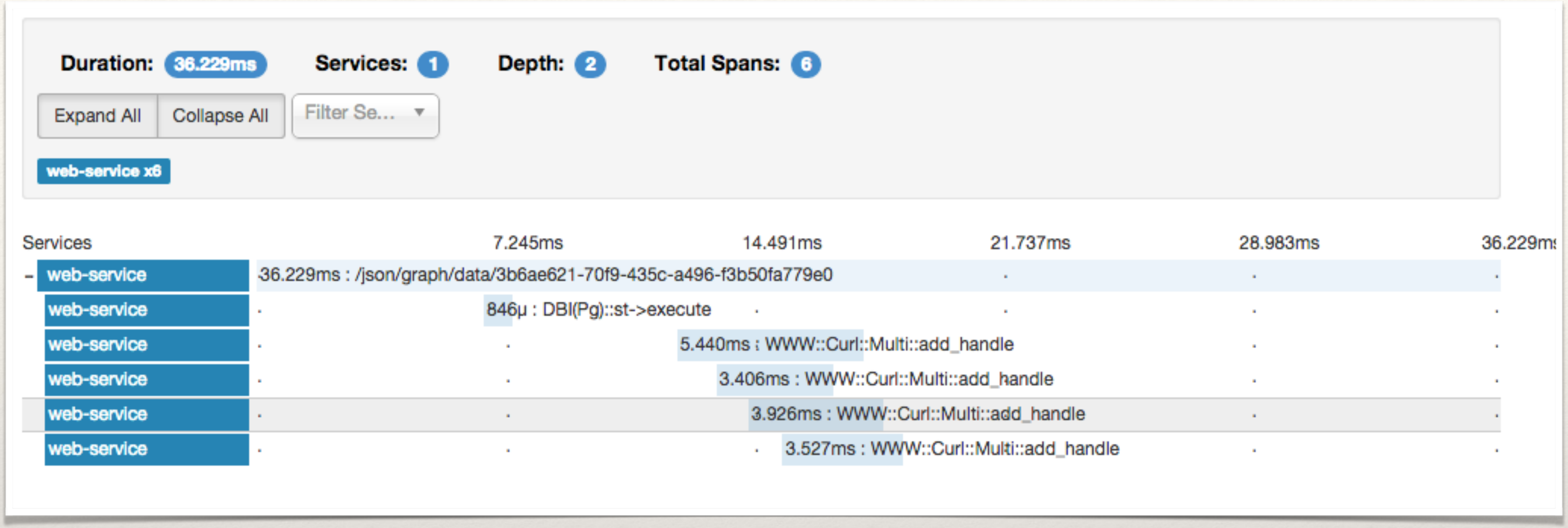


Telling computers what to do.

Zipkin is Java/Scala

Wrote C support:
github.com/circonus-labs/libmtev

Wrote Perl support:
github.com/circonus-labs/circonus-tracer-perl



Duration: 36.229ms Services: 1 Depth: 2 Total Spans: 6

```
Services                 7.245ms        14.491ms        21.737ms        28.983ms        36.229ms
- web-service   36.229ms : /json/graph/data/3b6ae621-70f9-435c-a496-f3b50fa779e0
  web-service            846µ : DBI(Pg)::st->execute
  web-service                  5.440ms : WWW::Curl::Multi::add_handle
  web-service                    3.406ms : WWW::Curl::Multi::add_handle
  web-service                      3.926ms : WWW::Curl::Multi::add_handle
  web-service                        3.527ms : WWW::Curl::Multi::add_handle
```

A sample trace: data from S_2

Celebration

Day 1

Noticed unexpected topology queries.

Found a data location caching issue.

Shaved 350ms off every graph request.

Celebration

Day 4-7

Noticed frequent 150ms stalls in internal REST.

Frequent == 90%+

Found a libcurl issue (async resolver).

Shaved 150ms*(n*0.9) off ~50% of page loads.

Path 2

Tooling must expose fundamental systems behavior.

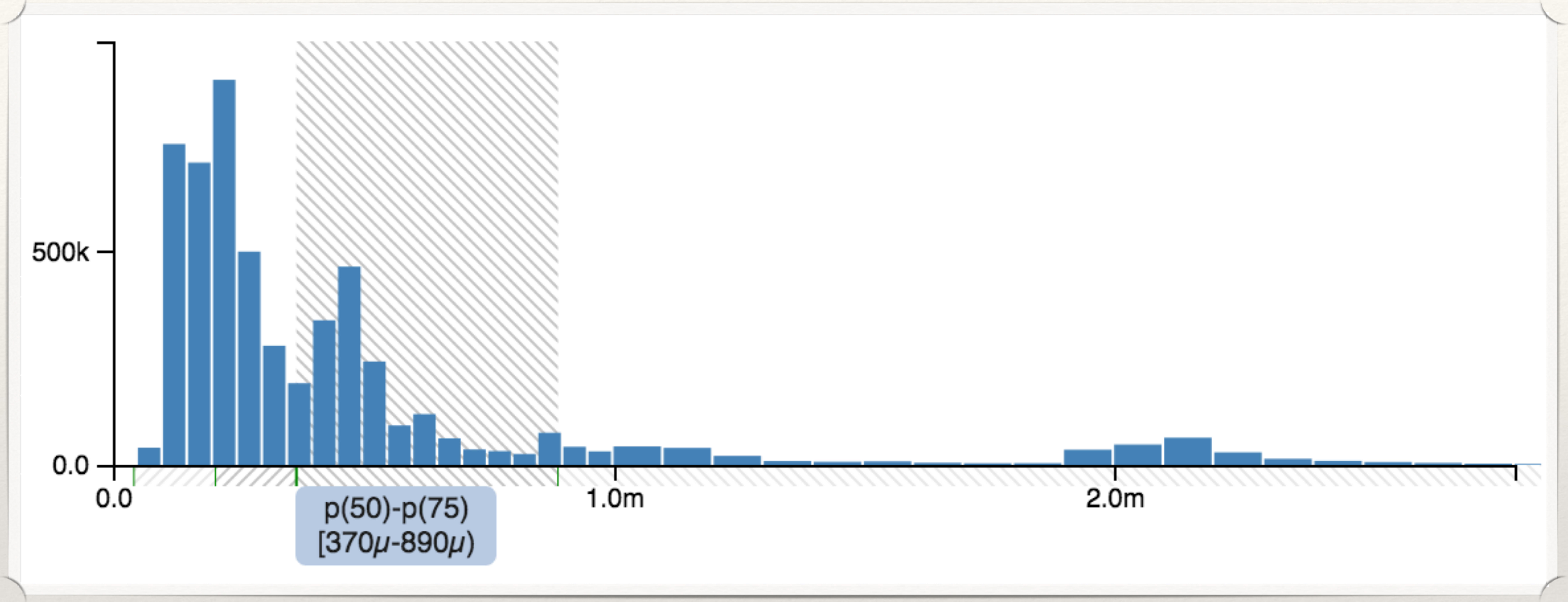
Sampling frequencies need to change.

First some statistical realities

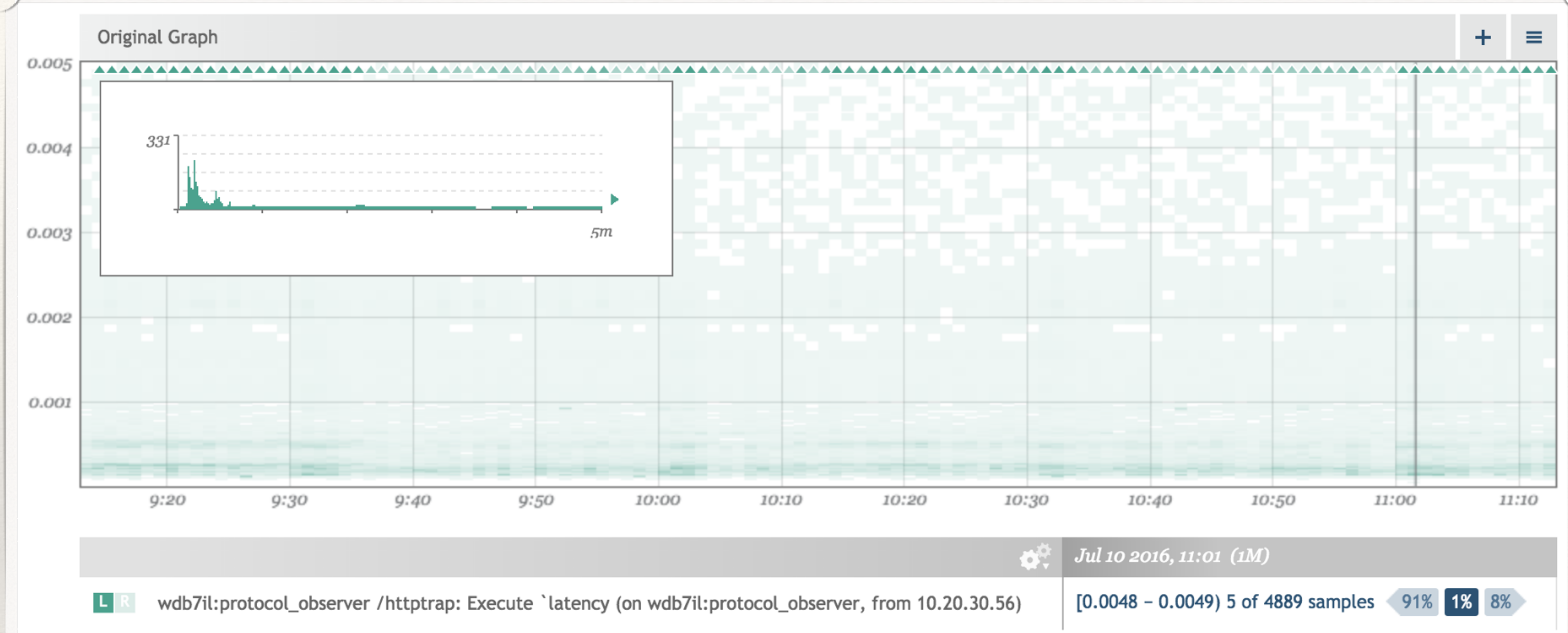
If your model has outliers; and most do.

It is rare that you can confidently claim a change in behavior from a single datapoint.

You need a lot of data.



At high volume,
understanding distributions well is the best we can do…
at least today.

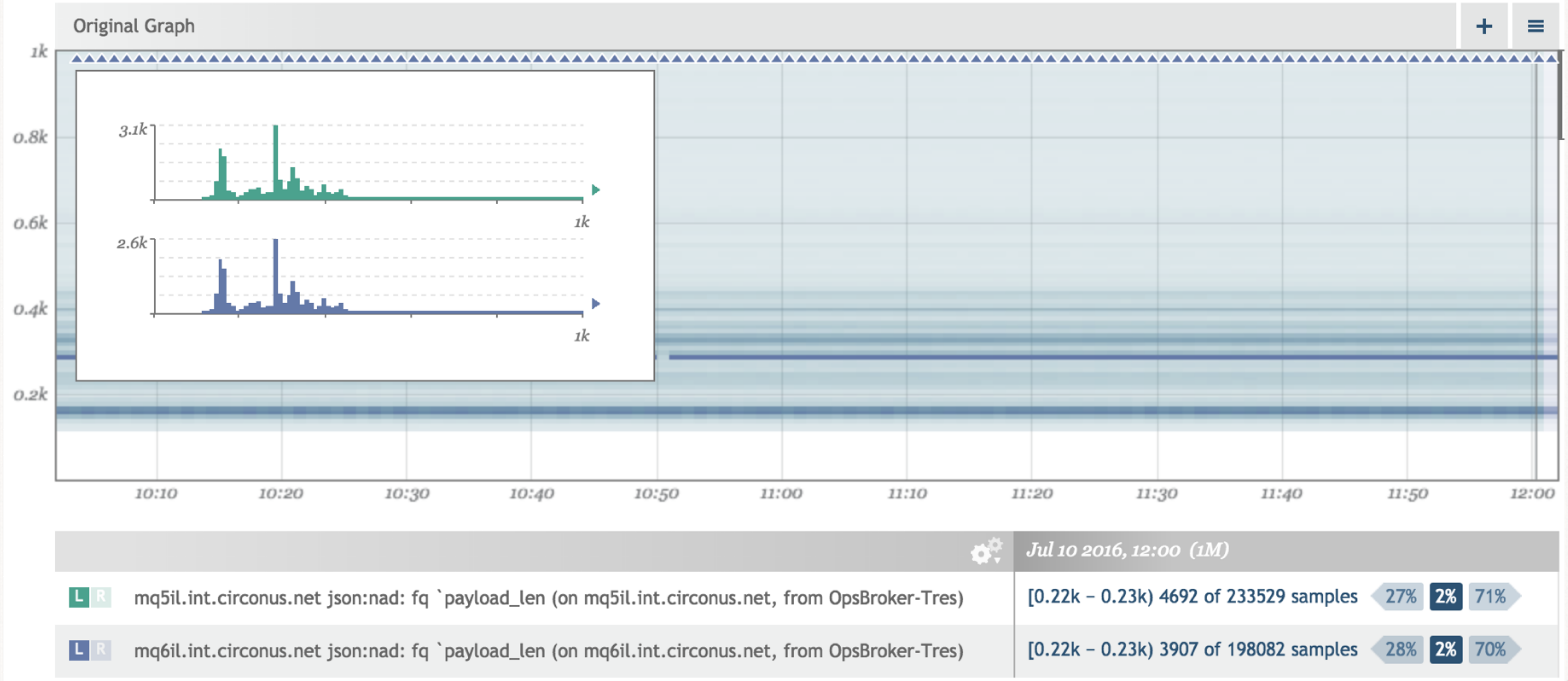


Jul 10 2016, 11:01 (1M)

L R wdb7il:protocol_observer /httptrap: Execute `latency (on wdb7il:protocol_observer, from 10.20.30.56)

[0.0048 – 0.0049) 5 of 4889 samples 91% 1% 8%

In order to model a system, you need to observe it correctly.



Jul 10 2016, 12:00 (1M)

L R	mq5il.int.circonus.net json:nad: fq `payload_len (on mq5il.int.circonus.net, from OpsBroker-Tres)	[0.22k – 0.23k) 4692 of 233529 samples 27% 2% 71%
L R	mq6il.int.circonus.net json:nad: fq `payload_len (on mq6il.int.circonus.net, from OpsBroker-Tres)	[0.22k – 0.23k) 3907 of 198082 samples 28% 2% 70%

A more concise model of behavior is required.

Original Graph
1k
0.8k
0.6k
0.4k
0.2k
2.8M
2.4M
May 16
May 24
Jun 1
Jun 8
Jun 16
Jun 23
Jul 1
Jul 8
Jun 30 2016, 00:00 (6h)

mq5il.int.circonus.net json:nad: fq `payload_len (on mq5il.int.circonus.net, from OpsBroker-Tres)	[0.28k – 0.29k) 15856112 of 130856646 samples 29% 12% 59%
mq6il.int.circonus.net json:nad: fq `payload_len (on mq6il.int.circonus.net, from OpsBroker-Tres)	[0.28k – 0.29k) 13549952 of 110974332 samples 29% 12% 59%

Because analysis of ~~240MM data points~~.
45 billion data points changes the scope.

Thanks!