Three Pillars with Zero Answers

A New Observability Scorecard

November 5, 2018

LIGHTSTEP

First, a Critique

The Conventional Wisdom

Observing microservices is hard

Google and Facebook solved this (right???)

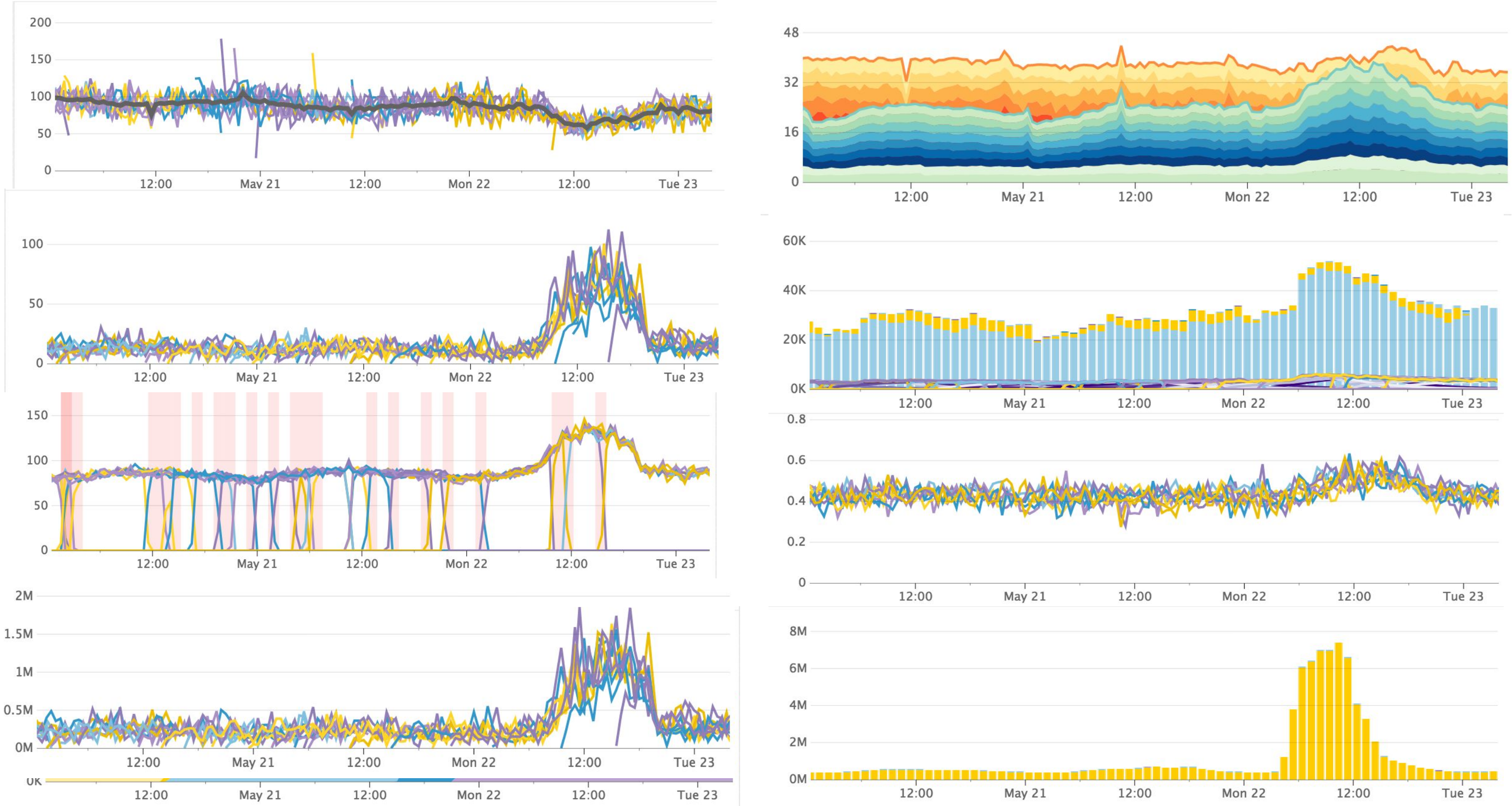
They used **Metrics, Logging, and Distributed Tracing…**

So we should, too.

The Three Pillars of Observability

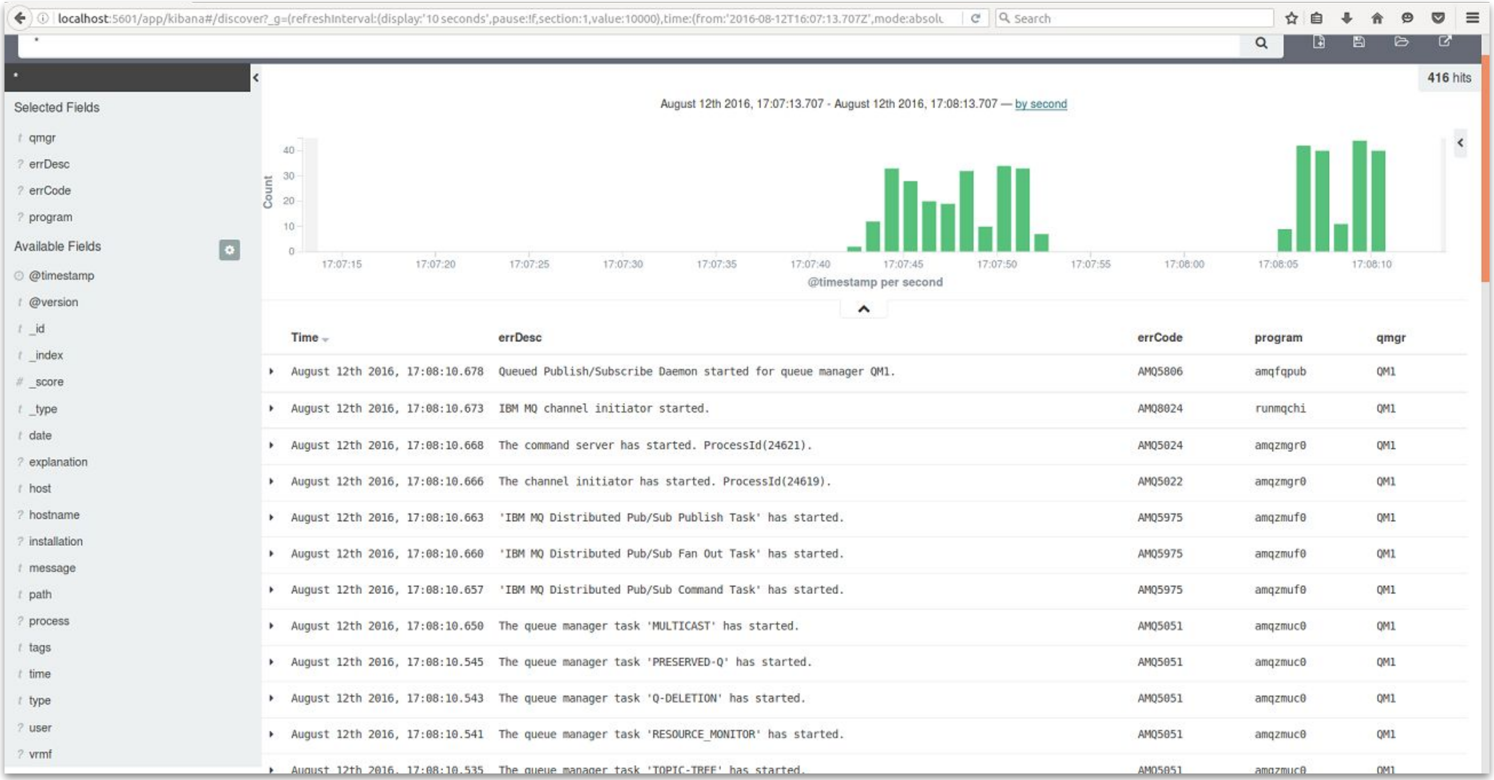
- Metrics
- Logging
- Distributed Tracing

Metrics!



200
150
100
50
0
12:00
May 21
12:00
Mon 22
12:00
Tue 23
100
50
0
150
100
50
0
2M
1.5M
1M
0.5M
0M
0K
48
32
16
0
60K
40K
20K
0K
0.8
0.6
0.4
0.2
0
8M
6M
4M
2M
0M

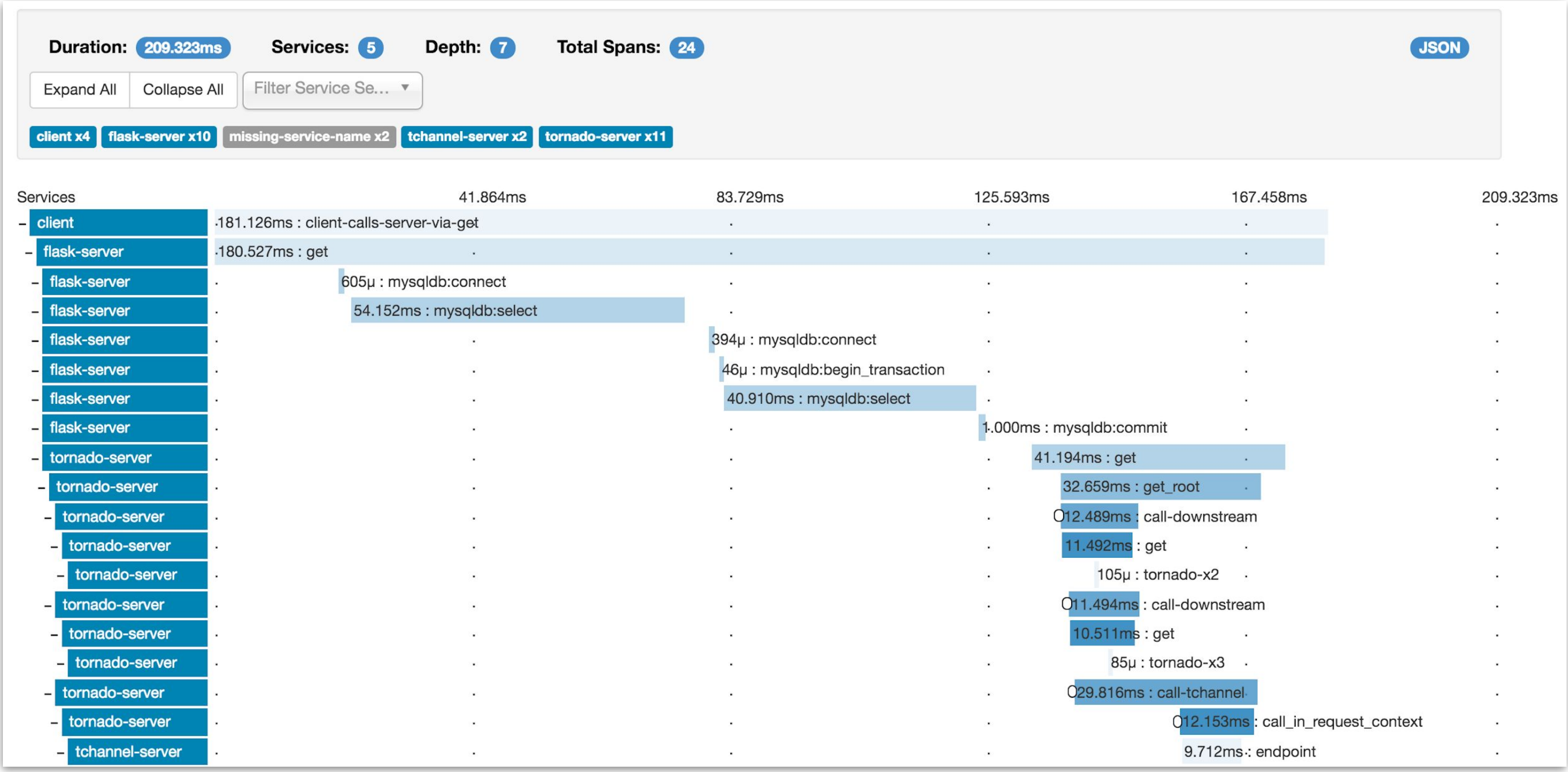
Logging!



August 12th 2016, 17:07:13.707 - August 12th 2016, 17:08:13.707 — by second

Time	errDesc	errCode	program	qmgr
August 12th 2016, 17:08:10.678	Queued Publish/Subscribe Daemon started for queue manager QM1.	AMQ5806	amqfqpub	QM1
August 12th 2016, 17:08:10.673	IBM MQ channel initiator started.	AMQ8024	runmqchi	QM1
August 12th 2016, 17:08:10.668	The command server has started. ProcessId(24621).	AMQ5024	amqzmgr0	QM1
August 12th 2016, 17:08:10.666	The channel initiator has started. ProcessId(24619).	AMQ5022	amqzmgr0	QM1
August 12th 2016, 17:08:10.663	'IBM MQ Distributed Pub/Sub Publish Task' has started.	AMQ5975	amqzmuf0	QM1
August 12th 2016, 17:08:10.660	'IBM MQ Distributed Pub/Sub Fan Out Task' has started.	AMQ5975	amqzmuf0	QM1
August 12th 2016, 17:08:10.657	'IBM MQ Distributed Pub/Sub Command Task' has started.	AMQ5975	amqzmuf0	QM1
August 12th 2016, 17:08:10.650	The queue manager task 'MULTICAST' has started.	AMQ5051	amqzmuc0	QM1
August 12th 2016, 17:08:10.545	The queue manager task 'PRESERVED-Q' has started.	AMQ5051	amqzmuc0	QM1
August 12th 2016, 17:08:10.543	The queue manager task 'Q-DELETION' has started.	AMQ5051	amqzmuc0	QM1
August 12th 2016, 17:08:10.541	The queue manager task 'RESOURCE_MONITOR' has started.	AMQ5051	amqzmuc0	QM1
August 12th 2016, 17:08:10.535	The queue manager task 'TOPIC-TREE' has started.	AMQ5051	amqzmuc0	QM1

Tracing!

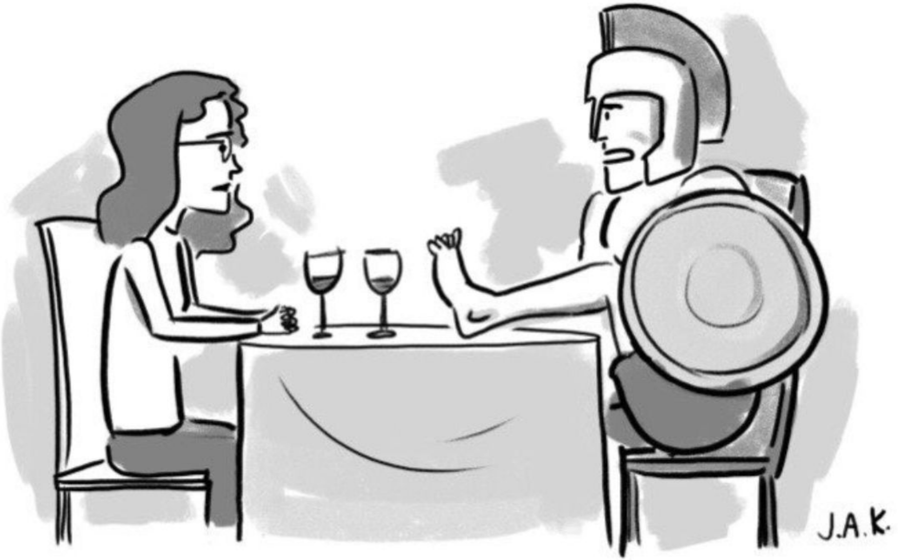


Duration: 209.323ms **Services:** 5 **Depth:** 7 **Total Spans:** 24

Services	
– client	181.126ms : client-calls-server-via-get
– flask-server	180.527ms : get
– flask-server	605µ : mysqldb:connect
– flask-server	54.152ms : mysqldb:select
– flask-server	394µ : mysqldb:connect
– flask-server	46µ : mysqldb:begin_transaction
– flask-server	40.910ms : mysqldb:select
– flask-server	1.000ms : mysqldb:commit
– tornado-server	41.194ms : get
– tornado-server	32.659ms : get_root
– tornado-server	12.489ms : call-downstream
– tornado-server	11.492ms : get
– tornado-server	105µ : tornado-x2
– tornado-server	11.494ms : call-downstream
– tornado-server	10.511ms : get
– tornado-server	85µ : tornado-x3
– tornado-server	29.816ms : call-tchannel
– tornado-server	12.153ms : call_in_request_context
– tchannel-server	9.712ms : endpoint

POP
SNAP
CRACKLE
CRACKLE!
POP!

Fatal Flaws

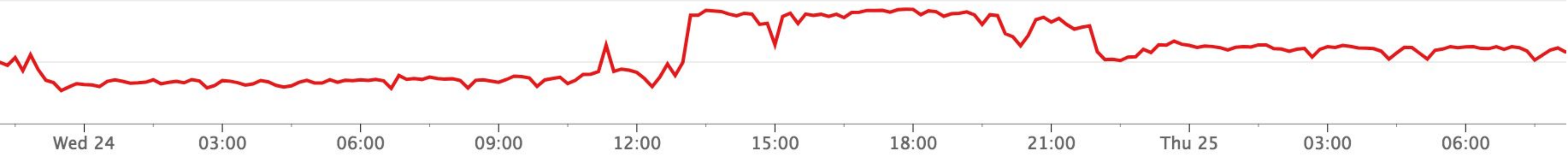


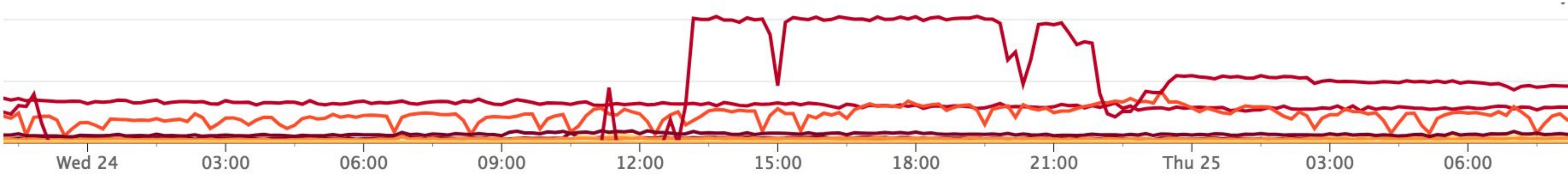
"I'm ready to be vulnerable."

A word nobody knew in 2015…

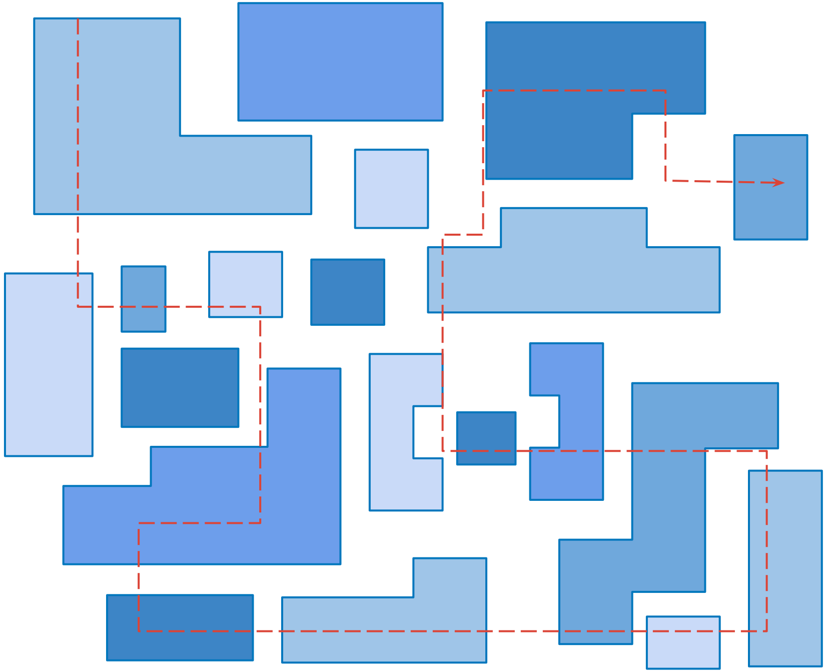
Dimensions (aka “tags”) can explain variance in timeseries data (aka “metrics”) …

… but **cardinality**





Logging Data Volume: a reality check



```
        transaction rate
x      all microservices
x   cost of net+storage
x     weeks of retention
---------------------------
      way too much $$$$
```

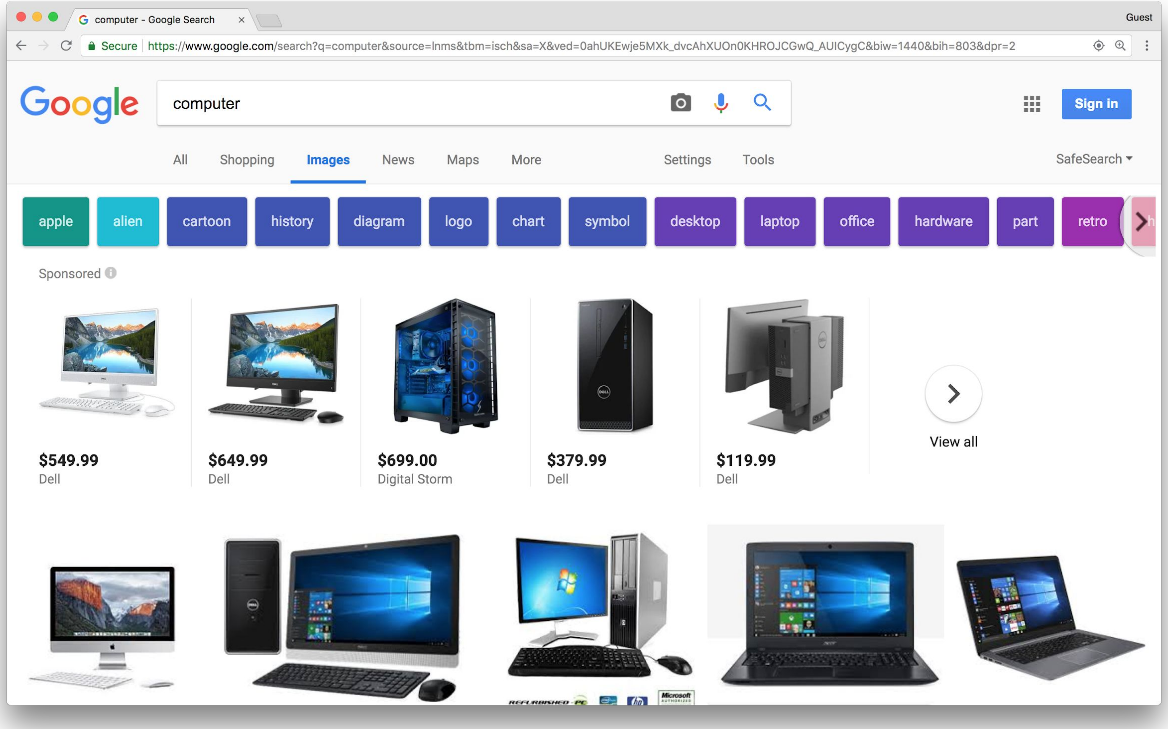
The Life of Transaction Data: Dapper

Stage	Overhead affects…	Retained
Instrumentation Executed	App	100.00%
Buffered within app process	App	000.10%
Flushed out of process	App	000.10%
Centralized regionally	Regional network + storage	000.10%
Centralized globally	WAN + storage	000.01%

Fatal Flaws

	Logs	Metrics	Dist. Traces
TCO scales gracefully	—	✓	✓
Accounts for all data (i.e., unsampled)	✓	✓	—
Immune to cardinality	✓	—	✓

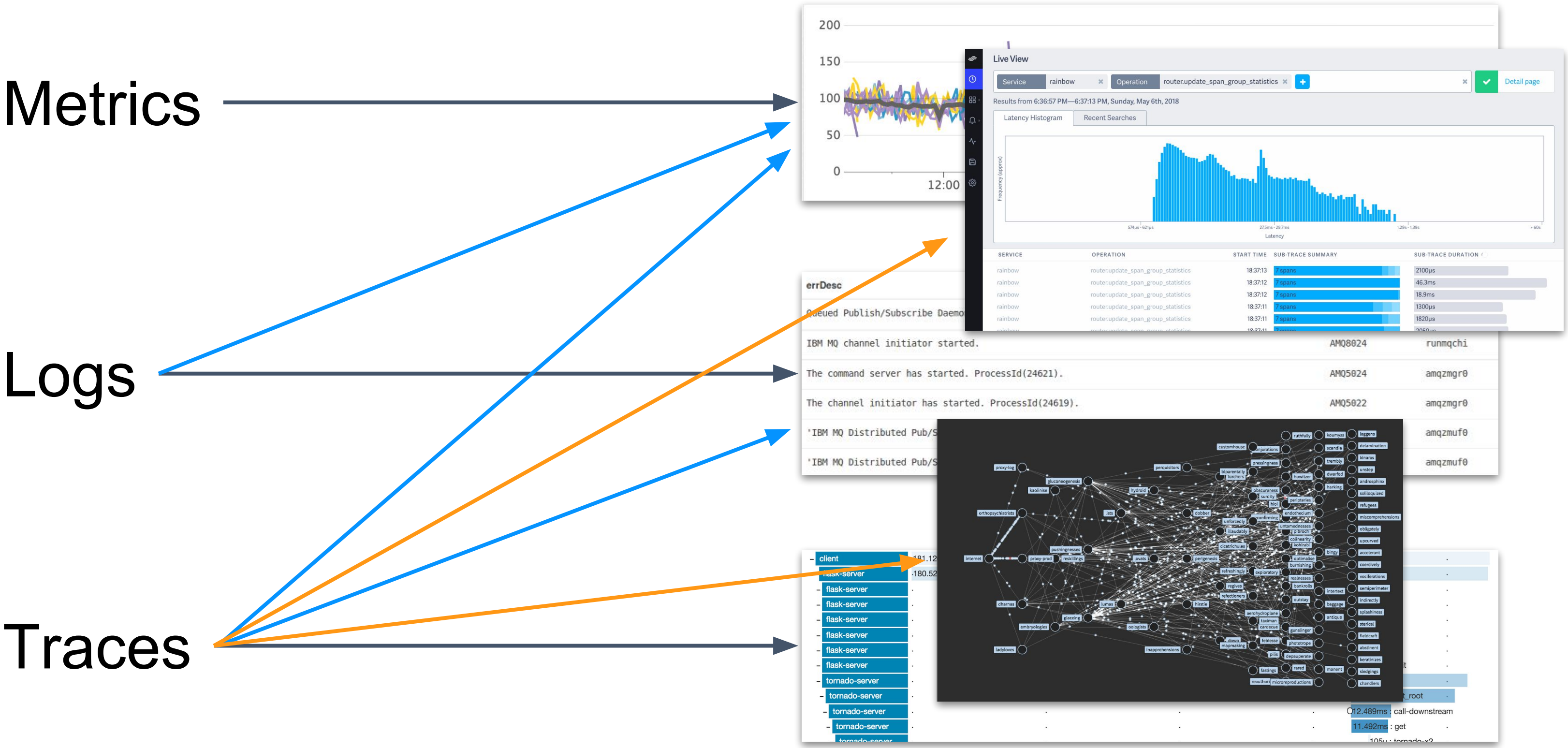
Data vs UI



https://www.google.com/search?q=computer&source=lnms&tbm=isch&sa=X&ved=0ahUKEwje5MXk_dvcAhXUOn0KHROJCGwQ_AUICygC&biw=1440&bih=803&dpr=2

computer

Data vs UI



Metrics

Logs

Traces

Live View

Service: rainbow ✖ Operation: router.update_span_group_statistics ✖ + ✖ ✓ Detail page

Results from 6:36:57 PM—6:37:13 PM, Sunday, May 6th, 2018

Latency Histogram | Recent Searches

SERVICE	OPERATION	START TIME	SUB-TRACE SUMMARY	SUB-TRACE DURATION
rainbow	router.update_span_group_statistics	18:37:13	7 spans	2100μs
rainbow	router.update_span_group_statistics	18:37:12	7 spans	46.3ms
rainbow	router.update_span_group_statistics	18:37:12	7 spans	18.9ms
rainbow	router.update_span_group_statistics	18:37:11	7 spans	1300μs
rainbow	router.update_span_group_statistics	18:37:11	7 spans	1820μs

```
errDesc
Queued Publish/Subscribe Daemo
IBM MQ channel initiator started.                      AMQ8024   runmqchi
The command server has started. ProcessId(24621).      AMQ5024   amqzmgr0
The channel initiator has started. ProcessId(24619).   AMQ5022   amqzmgr0
'IBM MQ Distributed Pub/S                                        amqzmuf0
'IBM MQ Distributed Pub/S                                        amqzmuf0
```

```
- client         181.12
  flask-server   180.52
  - flask-server
  - flask-server
  - flask-server
  - flask-server
  - flask-server
  - flask-server
  - tornado-server
    - tornado-server
      - tornado-server      12.489ms : call-downstream
        - tornado-server    11.492ms : get
          tornado-server    105μ : tornado-x2
```

Metrics, Logs, and Traces are **Just Data,**

… not a feature or use case.

A New Scorecard for Observability

Observability: Quick Vocab Refresher

“SLI” = “Service Level Indicator”

TL;DR: An SLI is **an indicator of health** that a service’s **consumers** would care about.

… *not* an indicator of its inner workings

Observability: Two Fundamental **Goals**

- Gradually improving an SLI
- Rapidly restoring an SLI

days, weeks, months…

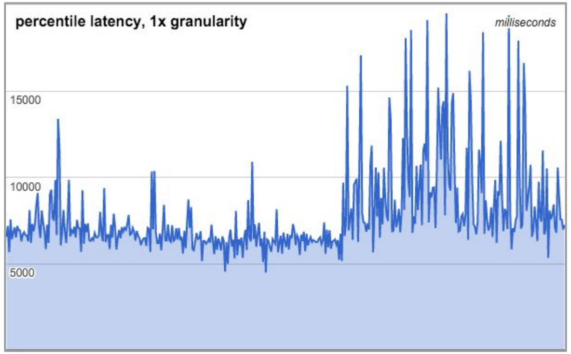
NOW!!!!

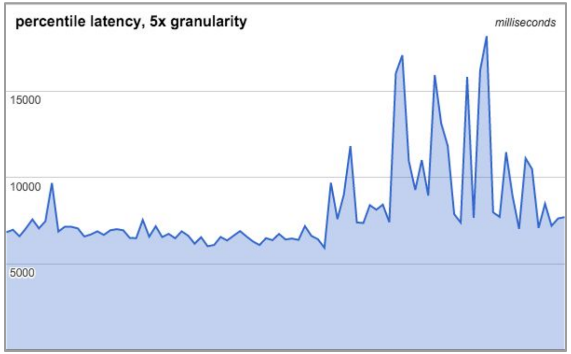
Reminder: “SLI” = “Service Level Indicator”

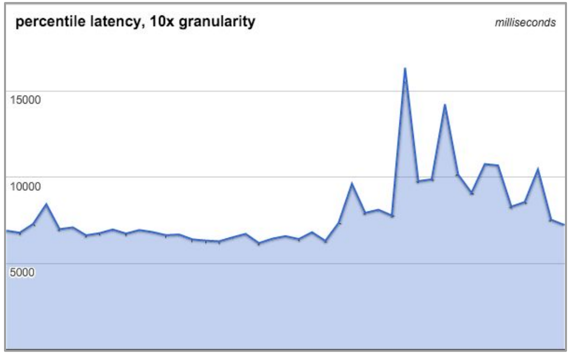
Observability: Two Fundamental **Activities**

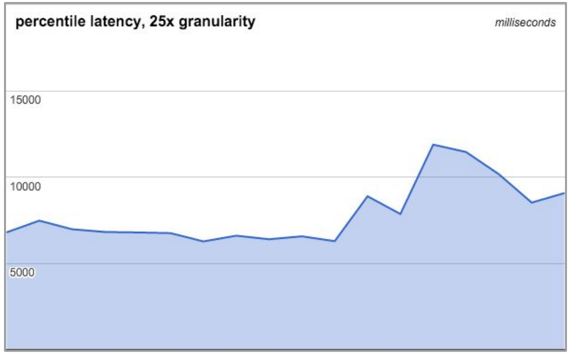
1. Detection: perfect SLI capture
2. Refinement: reduce the search space

An interlude about stats frequency









Scorecard >> **Detection**

Specificity:

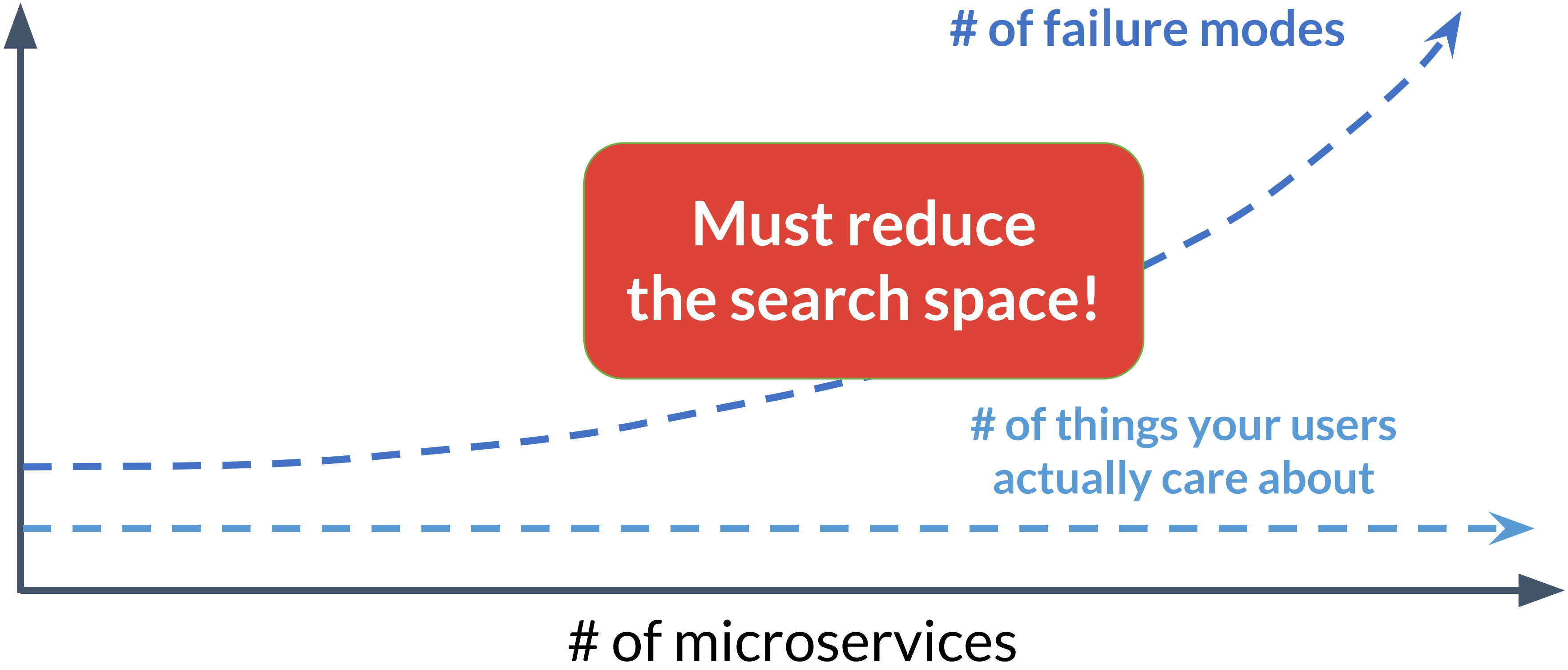
- Arbitrary dimensionality and cardinality
- Any layer of the stack, including mobile+web!

Fidelity:

- Correct stats!!!
- High stats frequency (i.e., “beware smoothing”!)

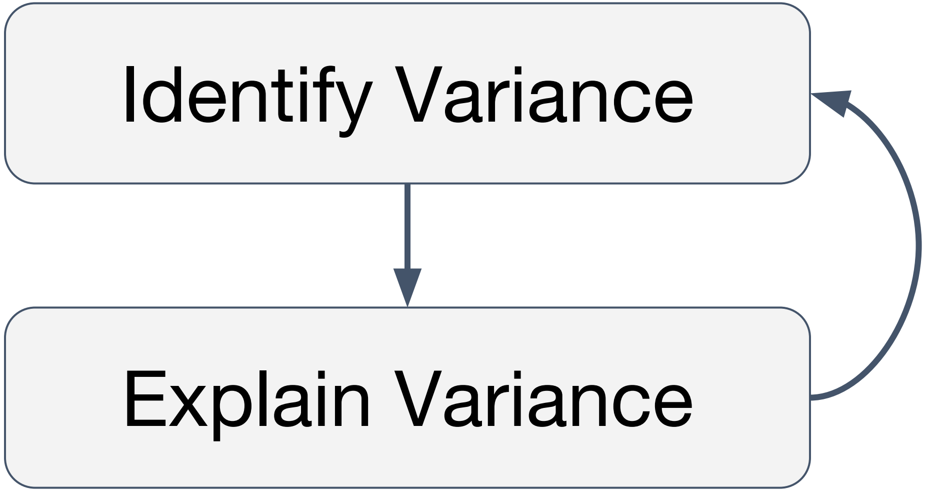
Freshness: ≤ 5 second lag

Scorecard >> **Refinement**



Must reduce the search space!

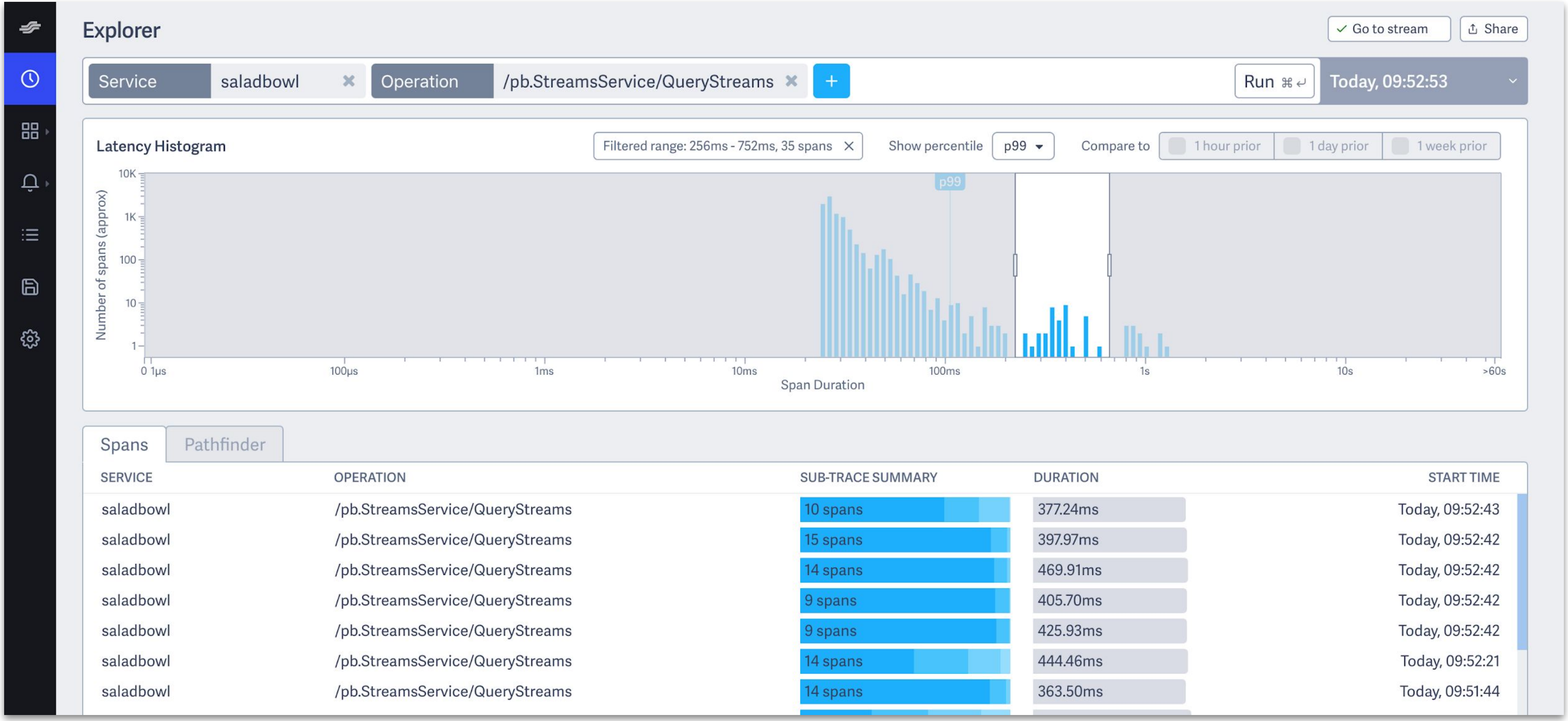
Scorecard >> **Refinement**



Identify Variance

Explain Variance

An interlude about variance and “p99”



Service: saladbowl ✖

Operation: /pb.StreamsService/QueryStreams ✖

Today, 09:52:53

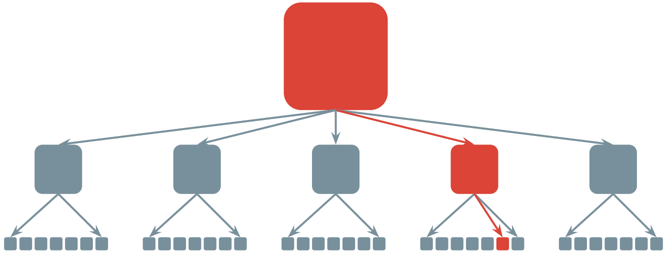
Filtered range: 256ms - 752ms, 35 spans ✕

Show percentile: p99

Compare to: □ 1 hour prior □ 1 day prior □ 1 week prior

SERVICE	OPERATION	SUB-TRACE SUMMARY	DURATION	START TIME
saladbowl	/pb.StreamsService/QueryStreams	10 spans	377.24ms	Today, 09:52:43
saladbowl	/pb.StreamsService/QueryStreams	15 spans	397.97ms	Today, 09:52:42
saladbowl	/pb.StreamsService/QueryStreams	14 spans	469.91ms	Today, 09:52:42
saladbowl	/pb.StreamsService/QueryStreams	9 spans	405.70ms	Today, 09:52:42
saladbowl	/pb.StreamsService/QueryStreams	9 spans	425.93ms	Today, 09:52:42
saladbowl	/pb.StreamsService/QueryStreams	14 spans	444.46ms	Today, 09:52:21
saladbowl	/pb.StreamsService/QueryStreams	14 spans	363.50ms	Today, 09:51:44

Scorecard >> **Refinement**



Identifying Variance:

- Cardinality: understand which tag changed
- Robust stats: *histograms* (see prev slide)
- Data retention: always “Know What’s Normal”

Explaining variance:

- Correct stats!!!
- “Suppress the messengers” of microservice failures

Wrapping up…

(first, a hint at my perspective)

(Review)

The Life of Transaction Data: Dapper

Stage	Overhead affects…	Retained
Instrumentation Executed	App	100.00%
Buffered within app process	App	000.10%
Flushed out of process	App	000.10%
Centralized regionally	Regional network + storage	000.10%
Centralized globally	WAN + storage	000.01%

The Life of Transaction Data: ~~Dapper~~ LightStep

Stage	Overhead affects…	Retained
Instrumentation Executed	App	100.00%
Buffered within app process	App	100.00%
Flushed out of process	App	100.00%
Centralized regionally	Regional network + storage	100.00%
Centralized globally	WAN + storage	on-demand

An Observability Scorecard

Detection

- Specificity: unlimited cardinality, across the entire stack
- Fidelity: correct stats, high stats frequency
- Freshness: ≤ 5 seconds

Refinement

- Identifying variance: unlimited cardinality, hi-fi histograms, data retention
- “Suppress the messengers”

Thank you!

Ben Sigelman, Co-founder and CEO
twitter: @el_bhs
email: bhs@lightstep.com

LIGHTSTEP