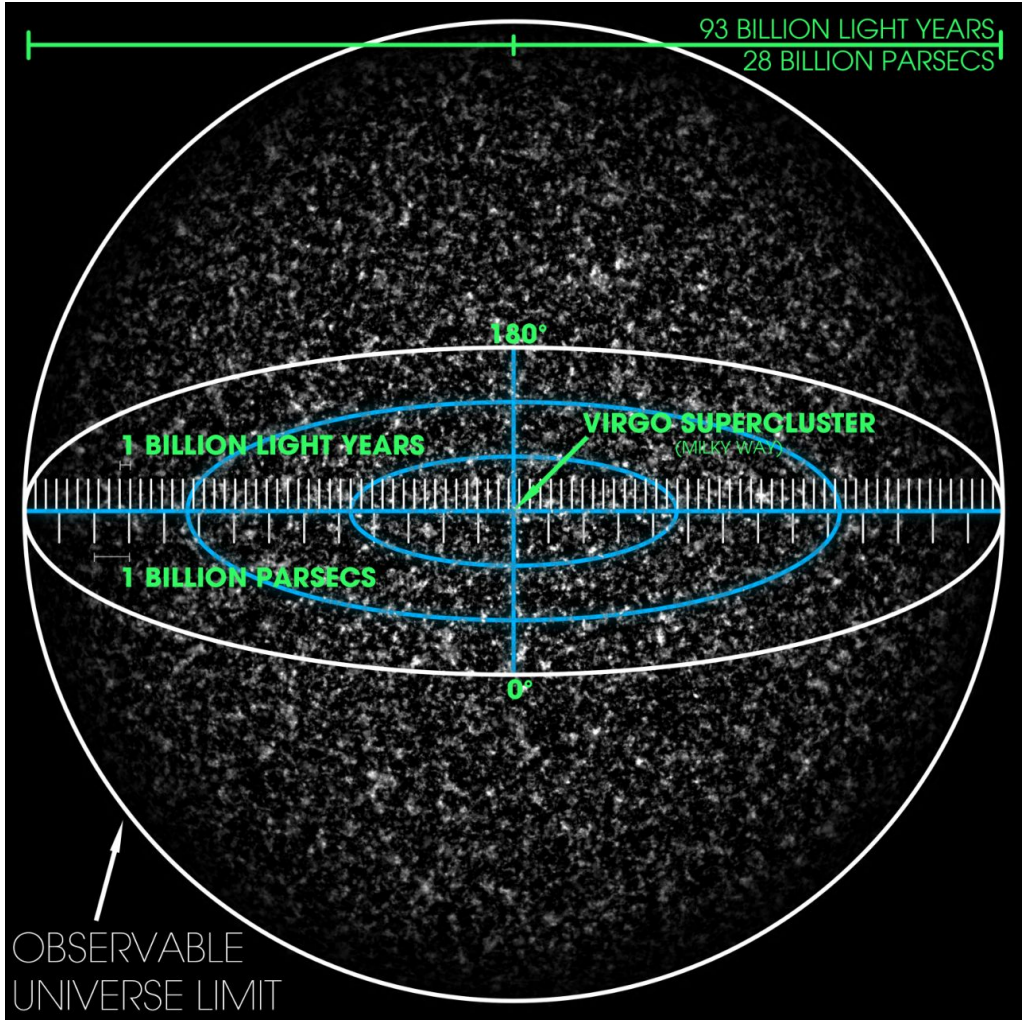
An Overview of Observability

v0.2.0


93 BILLION LIGHT YEARS
28 BILLION PARSECS
180°
VIRGO SUPERCLUSTER
(MILKY WAY)
1 BILLION LIGHT YEARS
1 BILLION PARSECS
0°
OBSERVABLE
UNIVERSE LIMIT

Definition

“In control theory, observability is the ability to understand what is going on in the inner workings of a system just by observing it from the outside.”

Translate!

Your software should explain itself and what is doing!

Three (+1) Pillars of Observability

- Logs
- Metrics
- Traces
- Events

Logs

Logs

- **Structured logging** vs. non-structured logging
- Structured logs can have any arbitrary shape and size.
- Usually used for debugging purposes and tracking down issues!
- Easy to implement.

Shortcomings

- Logs are very **expensive** at scale!
- They cannot be used for real-time computational purposes.
- Logs are also hard to track across different and distributed processes.
- You need know what to look for ahead of the time (know unknowns vs. unknown unknowns).

Standards

- APIs: **JSON** logging
- Log Aggregators: **Fluentd**, Logstash, Filebeat
- Log Databases: **Elasticsearch**, ...
- Dashboards: **Kibana**, ...

Metrics

Metrics

- Metrics are **time-series** data (*regular*) with **low cardinality**.
- They are aggregated by time.
- Metrics are used for **real-time** monitoring purposes.
- Metrics are very good at taking the **distribution of data** into account.
- Using metrics, we can implement **SLI**s and **SLO**s.
- **Alerting** (on SLO violation) is possible with metrics.

Service-LeveL Indicators

- An SLI is a service level indicator—a carefully defined quantitative measure of some aspect of the level of service that is provided.
- Examples:
  - **request latency**
  - **system throughput**
  - **error rates**
  - **availability**
  - **durability**.

Service-LeveL Objectives

- An SLO is a target value or range of values for a service level that is measured by an SLI.
- Examples:
  - 99.9% (3 nines) of requests respond in 10ms or better.
  - 99.999 (5 nines) of requests are processed with 5Mb/s or better.

Shortcomings

- Metrics CANNOT be broken down by high-cardinality dimensions

  (unique ids such user ids and so forth).

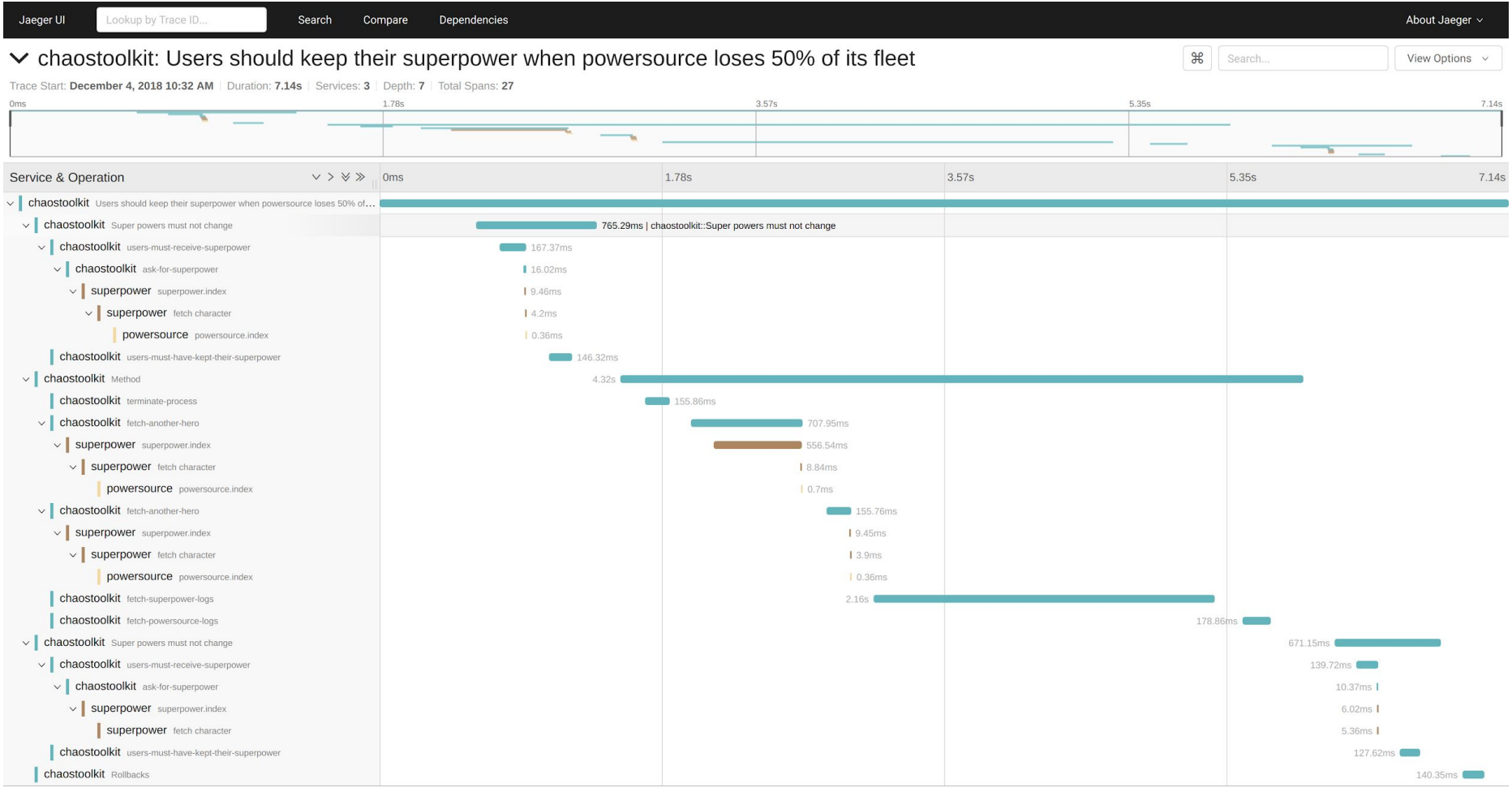
Standards

- APIs: **OpenMetrics**, ...
- Clients: **Prometheus**, ...
- Metrics Databases: **Prometheus**, ...
- Dashboards: **Grafana**, ...

Traces

Traces

- Traces are used for **debugging** and tracking requests across different processes and services.
- They can be used for identifying performance **bottlenecks**.


Jaeger UI
Lookup by Trace ID...
Search
Compare
Dependencies
About Jaeger
chaostoolkit: Users should keep their superpower when powersource loses 50% of its fleet
Search...
View Options
Trace Start: December 4, 2018 10:32 AM | Duration: 7.14s | Services: 3 | Depth: 7 | Total Spans: 27
0ms
1.78s
3.57s
5.35s
7.14s
Service & Operation
chaostoolkit Users should keep their superpower when powersource loses 50% of…
chaostoolkit Super powers must not change
765.29ms | chaostoolkit::Super powers must not change
chaostoolkit users-must-receive-superpower
167.37ms
chaostoolkit ask-for-superpower
16.02ms
superpower superpower.index
9.46ms
superpower fetch character
4.2ms
powersource powersource.index
0.36ms
chaostoolkit users-must-have-kept-their-superpower
146.32ms
chaostoolkit Method
4.32s
chaostoolkit terminate-process
155.86ms
chaostoolkit fetch-another-hero
707.95ms
superpower superpower.index
556.54ms
superpower fetch character
8.84ms
powersource powersource.index
0.7ms
chaostoolkit fetch-another-hero
155.76ms
superpower superpower.index
9.45ms
superpower fetch character
3.9ms
powersource powersource.index
0.36ms
chaostoolkit fetch-superpower-logs
2.16s
chaostoolkit fetch-powersource-logs
178.86ms
chaostoolkit Super powers must not change
671.15ms
chaostoolkit users-must-receive-superpower
139.72ms
chaostoolkit ask-for-superpower
10.37ms
superpower superpower.index
6.02ms
superpower fetch character
5.36ms
chaostoolkit users-must-have-kept-their-superpower
127.62ms
chaostoolkit Rollbacks
140.35ms

Shortcomings

- Due to their very data-heavy nature, traces need to be **sampled**.
- Tracing data are not optimized for aggregation.
- Due to sampling, we cannot precisely know about the distribution of data (detecting outliers is very hard).

Standards

- APIs: **OpenTracing**
- Implementations: **Jaeger**, ZipKin
- Dashboards: **Jaeger**, ...

Events

Events

- Events are **time-series** (*irregular*) data.
- They occur in temporal order, but the interval between occurrences are inconsistent and **sporadic**.
- Events are used for reporting and **alerting** on important or critical events such as errors, crashes, etc.

Shortcomings

- Very limited use cases

SaaS

- Rollbar
- Airbrake
- Sentry

Demo

DRY!

- Observing microservices is difficult!
- Microservices are about a lot of repeating yourself!
- Don’t repeat yourself!
- Core libraries should give us insights from inside out!

Rethinking Observability

Wait!

Does this sound right?

What’s Wrong?

- Logs, metrics, and traces each prematurely optimize one thing and comprise another thing based on a premise upfront.

You Don’t Want!

- You don’t want to write duplicate data into three different places.
- You don’t want to copy-paste IDs from tool to tool trying to track down a single problem!
- You don’t want to pay for three (four) different services doing almost the same thing!

You Want!

- You want one source of truth for your observability data.
- You want to be able to look at high-level dashboards, spot anomalies, and zoom in to get detailed information as needed.
- You want your observability cost be 10% to 30% of your total infrastructure cost.

Can We Keep All The Data?

- It is not practical to keep all the data!
- You are either throwing away data at ingestion time by **aggregating** or you are throwing away data after that by **sampling**.
- Observability can be incredibly cost-effective by using intelligent sampling.

Solutions

- LightStep (https://lightstep.com)
- Honeycomb (https://www.honeycomb.io)
- Elastic APM (https://www.elastic.co/products/apm)

More Resources

- https://www.youtube.com/watch?v=EJV_CgiqIOE
- https://www.youtube.com/watch?v=8u8A-bhhlSg
- https://www.infoq.com/presentations/google-microservices

Questions?