Tracing vol 2023

Jaroslav Bohmer

What I will be talking about

- OTel ecosystem in last year
- Big companies learning expensive lessons
- How and why to build in house
- Future favours the brave, but let’s optimize work not done

OTel & friends in last year

- TAG OBS: Observability Whitepaper v1.0
- TAG OBS: OpenCost
- TAG OBS: Query Standardization WG
- OTel Collector: Connectors
- OTel Col Contrib
  - 72 -> 89 receivers
  - 43 -> 47 exporters
  - 22 -> 23 processors

TAG Observability

OTel ecosystem is stabilizing

We have lego bricks for almost everything

Beyond data flows and ingestion in 2023

- Exporters generally fall in 3 specific categories
    - Pipes
    - Datastores
    - Observability vendors pipes
- Datastores are key for in-house observability solutions (focus on tracing)
    - In 2023 we have support for
        - Open Search dashboards
            - opensearch-project/data-prepper
        - Grafana dashboards
            - Clickhouse-datasource -> March 2023
            - Tempo-datasource
            - Jaeger-datasource
        - Jaeger tracing dashboards
            - Cassandra
            - Elasticsearch
            - Clickhouse -> Design done, still in progress

Big company window

Coinbase -> Datadog $65M for observability for 2021

- For observability, Coinbase spun up a dedicated team with the goal of moving off of Datadog, and onto a Grafana/Prometheus/Clickhouse stack.
- Datadog last minute deal kept Coinbase on platform, but bill was nowhere close to 2021

Shopify -> Datadog deemed too expensive with context of 20% of employees let go

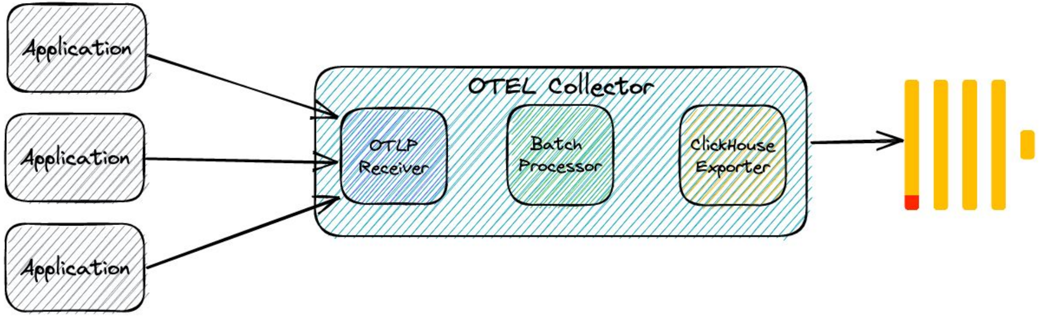
- Grafana/Prometheus solution , not sure about data storage

Uber -> 1.5 M datapoints per second is too much for any vendor

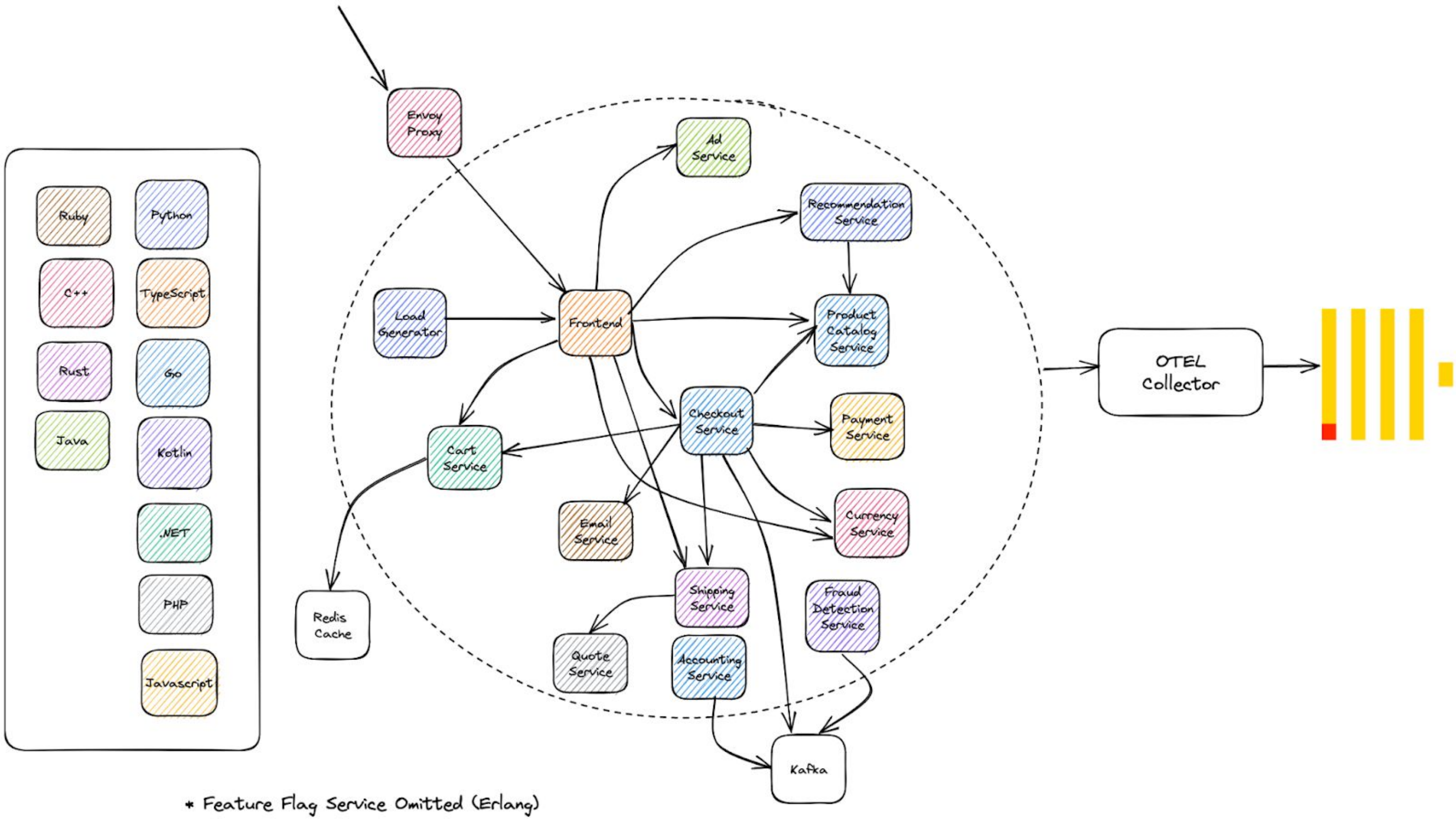
- Older example where Uber build M3 to scale prometheus beyond it’s design limits

Ok so we have lego bricks, but what we are building ?

- Datastore -> Clickhouse
- Dashboards -> Grafana Clickhouse datastore suite
- Exporter -> OTel clickhouse exporter
- Pipes & Instrumentation -> OTel collector and SDK

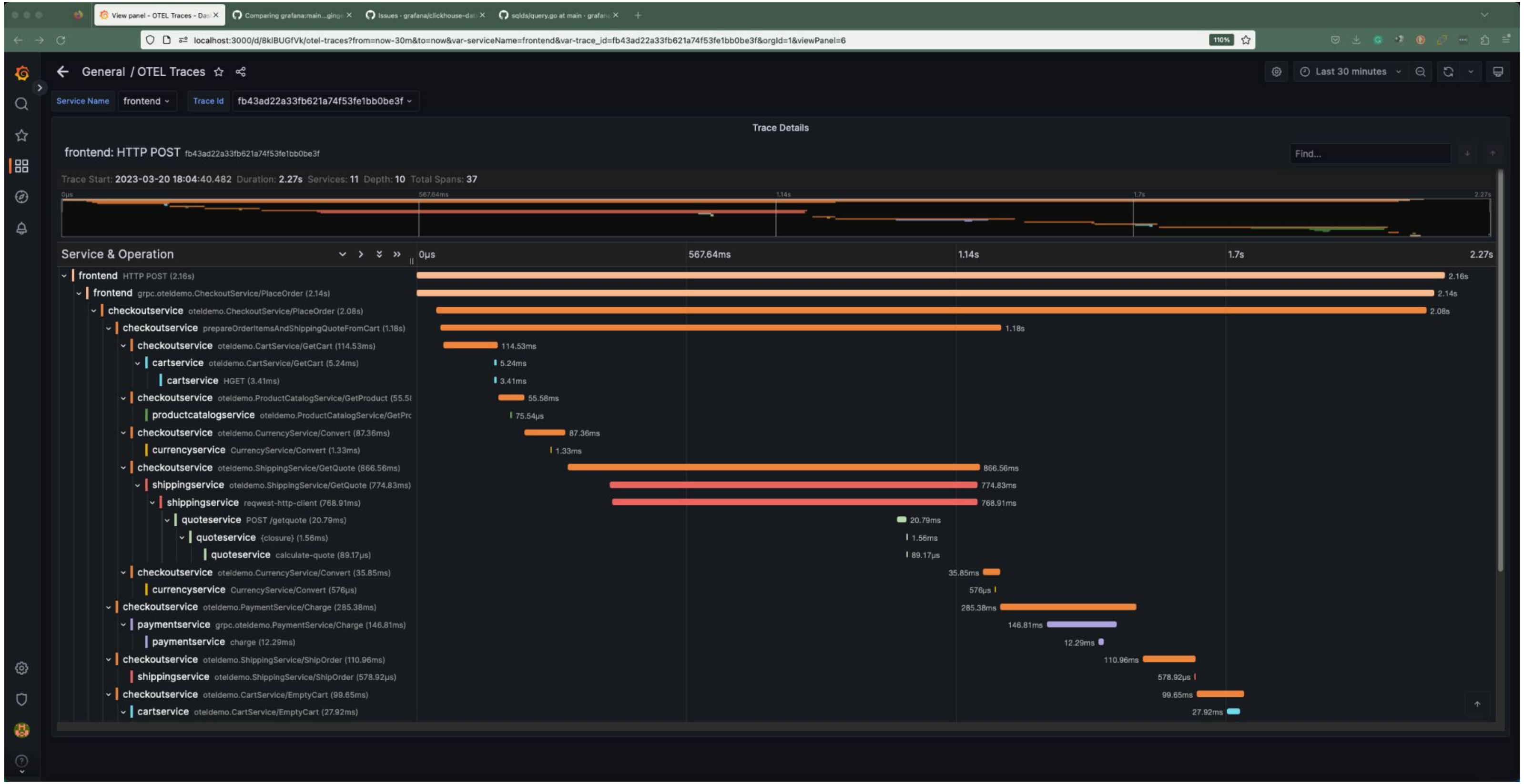


Classic OTel demo + clickhouse exporter



* Feature Flag Service Omitted (Erlang)

If Live demo fails



View panel - OTEL Traces - Das
Comparing grafana:main...ging
Issues · grafana/clickhouse-dat
sqlds/query.go at main · grafana
110%
Last 30 minutes
Find...
0μs
567.64ms
1.14s
1.7s
2.27s

localhost:3000/d/8kIBUGfVk/otel-traces?from=now-30m&to=now&var-serviceName=frontend&var-trace_id=fb43ad22a33fb621a74f53fe1bb0be3f&orgId=1&viewPanel=6

General / OTEL Traces

Service Name: frontend
Trace Id: fb43ad22a33fb621a74f53fe1bb0be3f

Trace Details

frontend: HTTP POST fb43ad22a33fb621a74f53fe1bb0be3f

Trace Start: 2023-03-20 18:04:40.482 Duration: 2.27s Services: 11 Depth: 10 Total Spans: 37

```
Service & Operation
frontend HTTP POST (2.16s)
  frontend grpc.oteldemo.CheckoutService/PlaceOrder (2.14s)
    checkoutservice oteldemo.CheckoutService/PlaceOrder (2.08s)
      checkoutservice prepareOrderItemsAndShippingQuoteFromCart (1.18s)
        checkoutservice oteldemo.CartService/GetCart (114.53ms)
          cartservice oteldemo.CartService/GetCart (5.24ms)
            cartservice HGET (3.41ms)
        checkoutservice oteldemo.ProductCatalogService/GetProduct (55.5
          productcatalogservice oteldemo.ProductCatalogService/GetPrc
        checkoutservice oteldemo.CurrencyService/Convert (87.36ms)
          currencyservice CurrencyService/Convert (1.33ms)
        checkoutservice oteldemo.ShippingService/GetQuote (866.56ms)
          shippingservice oteldemo.ShippingService/GetQuote (774.83ms)
            shippingservice reqwest-http-client (768.91ms)
              quoteservice POST /getquote (20.79ms)
                quoteservice {closure} (1.56ms)
                  quoteservice calculate-quote (89.17μs)
        checkoutservice oteldemo.CurrencyService/Convert (35.85ms)
          currencyservice CurrencyService/Convert (576μs)
      checkoutservice oteldemo.PaymentService/Charge (285.38ms)
        paymentservice grpc.oteldemo.PaymentService/Charge (146.81ms)
          paymentservice charge (12.29ms)
      checkoutservice oteldemo.ShippingService/ShipOrder (110.96ms)
        shippingservice oteldemo.ShippingService/ShipOrder (578.92μs)
      checkoutservice oteldemo.CartService/EmptyCart (99.65ms)
        cartservice oteldemo.CartService/EmptyCart (27.92ms)
```

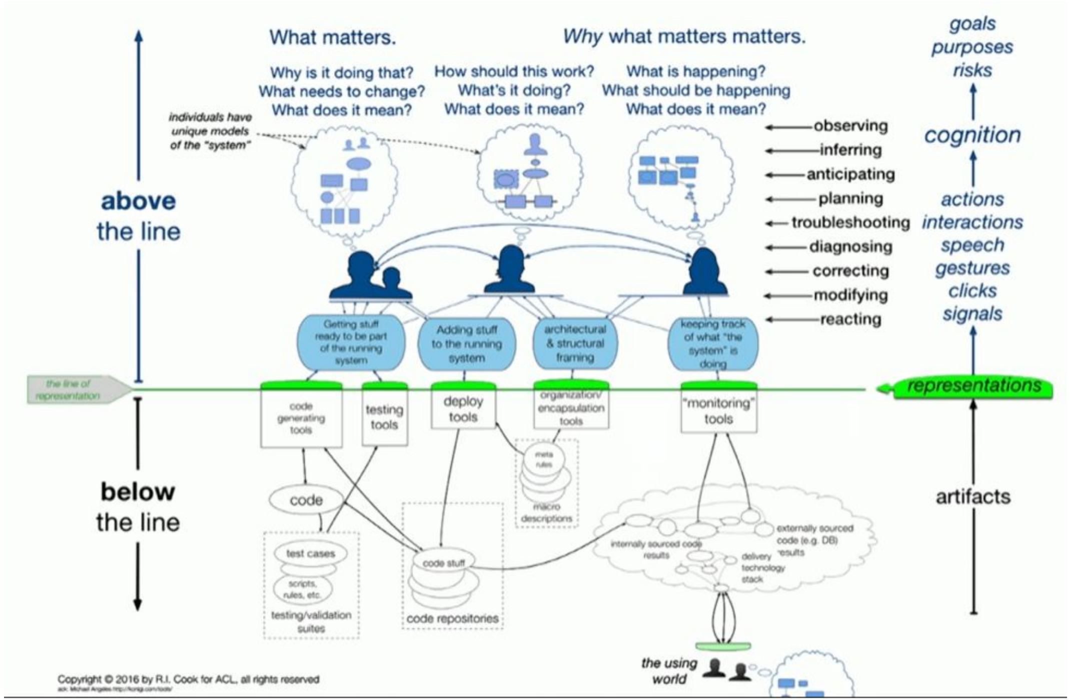
What we are missing compared to big players ?

- Default filtering option is just service
  - We need to customize dashboard and clickhouse indexes to get for example access to HTTP Agent filtering
- We don’t have nice dependency graphs
- Infered services, eg. (mysql server, latency breakdown per normalized query)
- RUM analytic dashboards are missing
- Metrification of Tracing data -> OTel Connectors maybe solution
  - Span Metric connector
- Resilience and production readiness

Ok so it’s worth it or not ? John Allspaw in 2017

As solution becomes more complicated, we need to work harder to see what is happening below the line.

If you recognizing “the line” as something existing in your company, then probably Observability and Open Telemetry are good topic to explore.



Copyright © 2016 by R.I. Cook for ACL, all rights reserved
ack: Michael Angeles http://konigi.com/tools/

Unfair advantage of in-house observability solutions

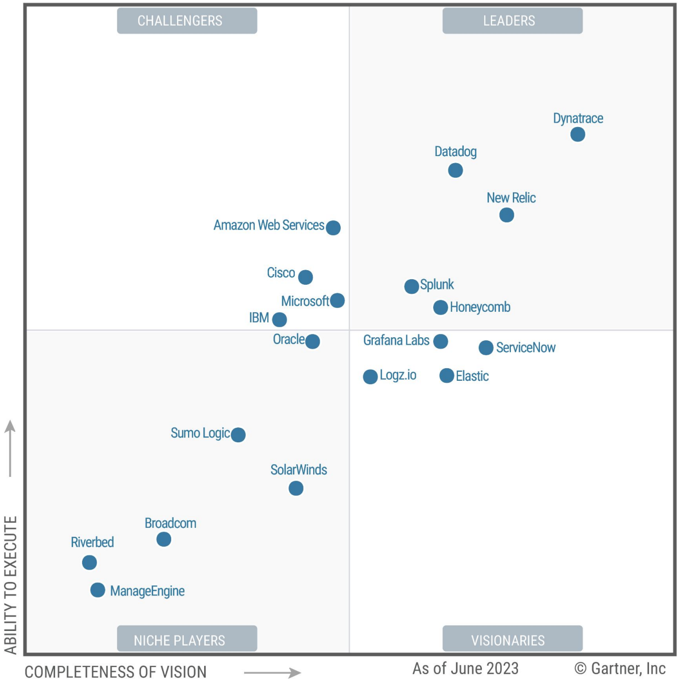
- Not all data is bringing same business value
    - Vendor without access to data cannot optimize solution according to that
        - When you emitting 150k spans per second, 11 TB of data per day this starts to matter
- Ok so let’s just sample data, easy right ?
    - Head sampling
        - In order to be viable you need to sample inside of context brackets (eg. API endpoints / workflows)
            - What about AWS region having problems ?
    - Tail sampling
        - You need to use “routing” at OTel collector level to ingest all spans information for 1 trace by 1 OTel collector to see if given trace is including error or anomaly
        - Memory needs of OTel collector raises significantly

Sampling is hard, persistence is easier

- I personally don’t think that sampling is good solution
- Having different persistence schedules should make storage requirement manageable
- Metrification of traces can help to catch trends & and with usage of exemplars you can select traces for long term storage
- Moving from transactional flows to streaming for ingestion can help with OTel Collector deployment design in terms of ingestion peaks

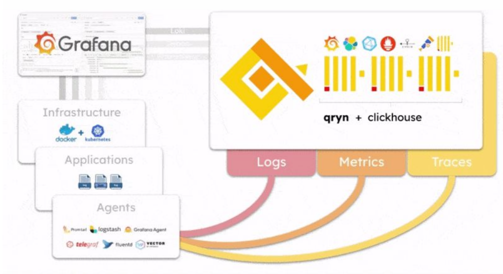
Playing field is highly competitive

Figure 1: Magic Quadrant for Application Performance Monitoring and Observability



Source: Gartner

SigNoz



Better Stack

Promise of Open telemetry

- Vendor lock-in problem got smaller
- New innovative companies are not afraid to take on established incumbents

But it didn’t change hard problems connected with observability

- A lot of data -> 10 kb of response can create 200 kb of span data
- Even if you have dashboards you need to work with data to get value

Good news at the end

Benchmarks:

TLDR: 120k -150k spans / s on 8 cpu box. So for most deployments you will need just 1 OTel collector + 1 clickhouse instance.

Jaeger and clickhouse

Clickhouse and observability

Trace based testing

Thank you