The State of SQL-Based Observability

Tanya Bragin
VP Product, ClickHouse

April 11, 2024

ClickHouse

ClickHouse

About me

- “Startup PM” that leans open-source
- Last 10-15 years in large scale data / telemetry use cases (ExtraHop, Elastic, ClickHouse)
- Have been both builder and user of observability tools and DIY stacks

What is SQL-Based Observability, and is it the right choice for me?

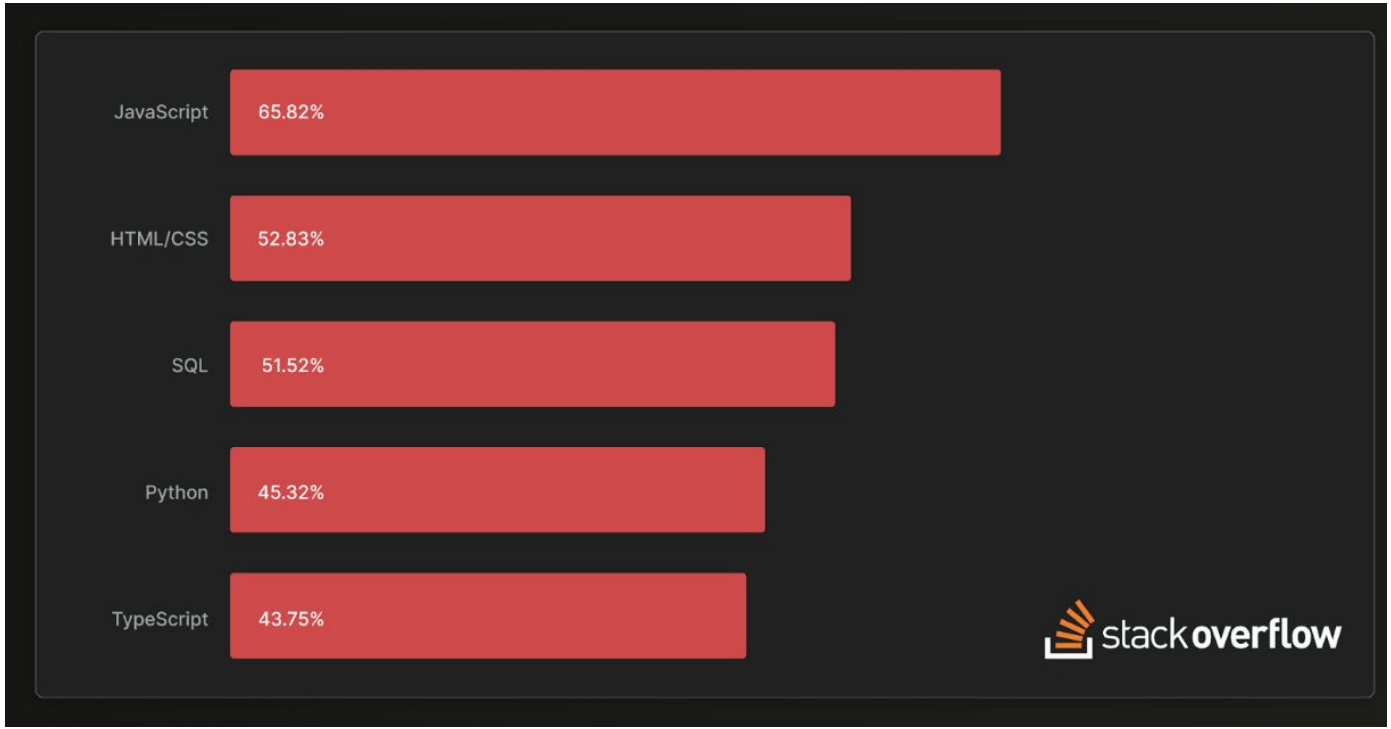
ClickHouse

Table of contents

Historical context

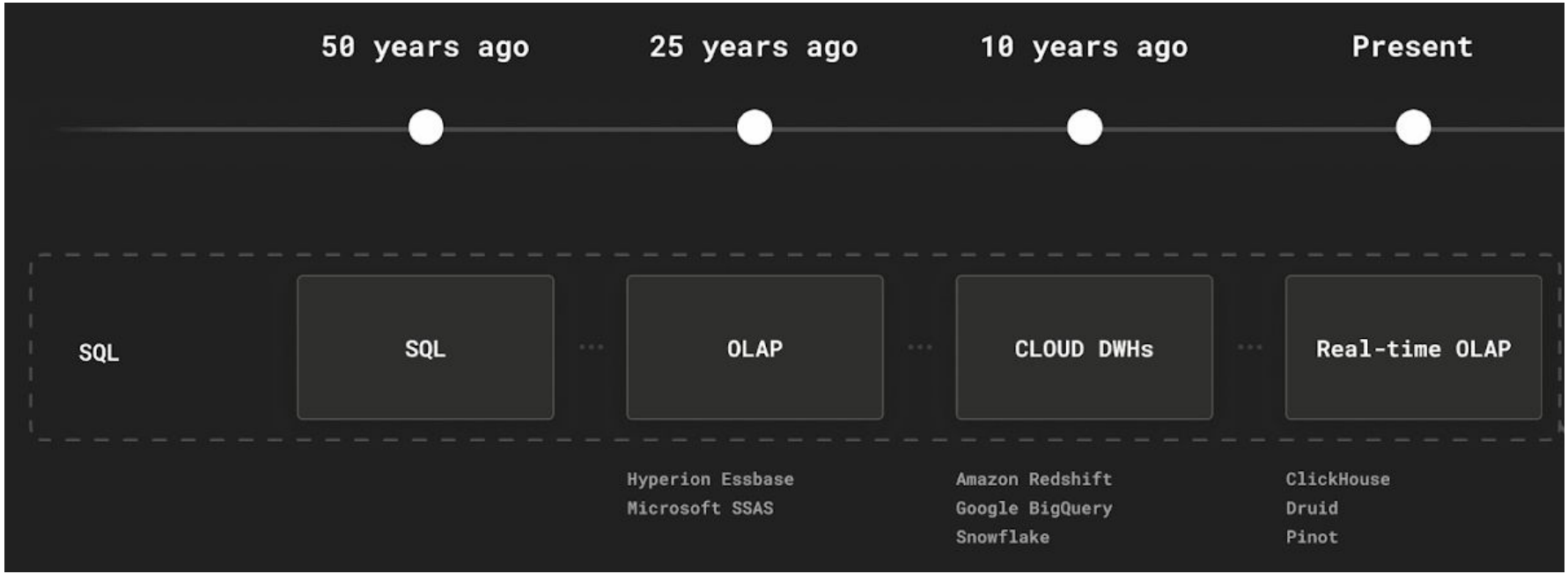
How did we get here?

SQL is 50 years old, and still popular

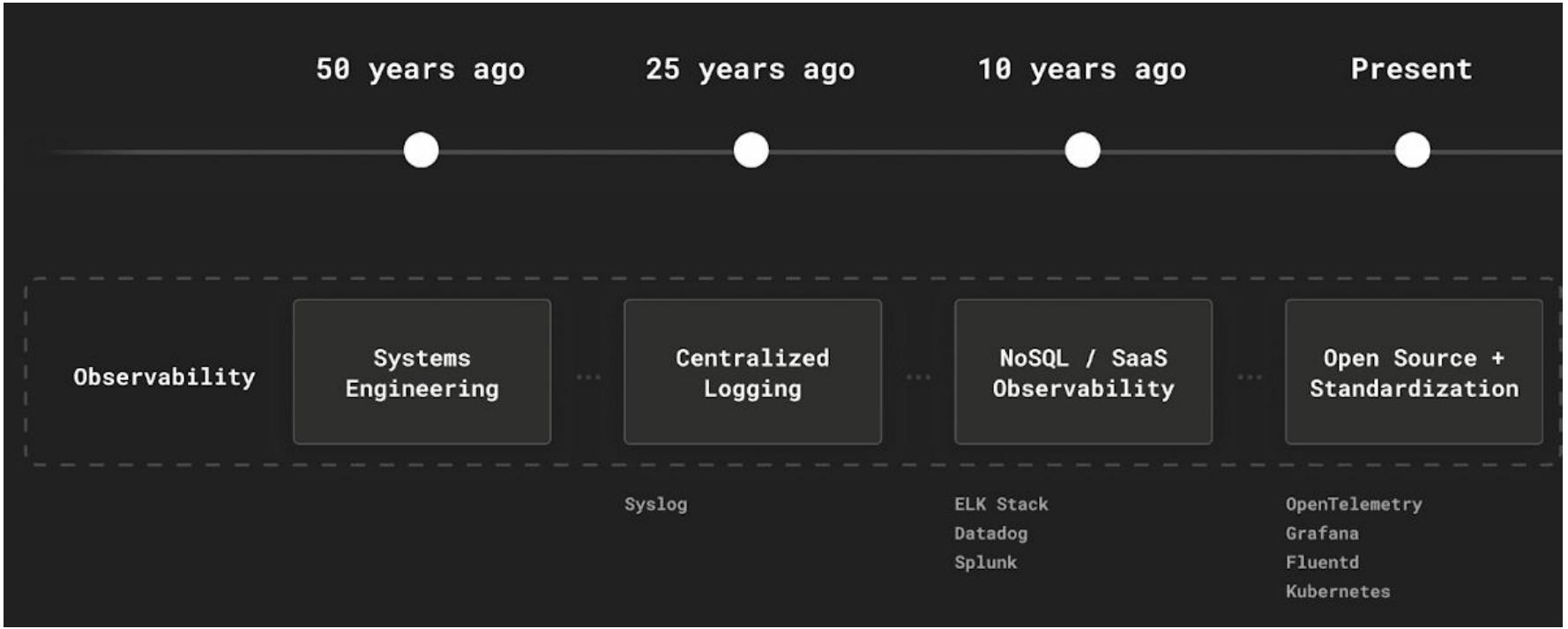


The 2023 StackOverflow Developer Survey ranked SQL as the 3rd most popular programming language, used by more than half of the 67K professional developers surveyed

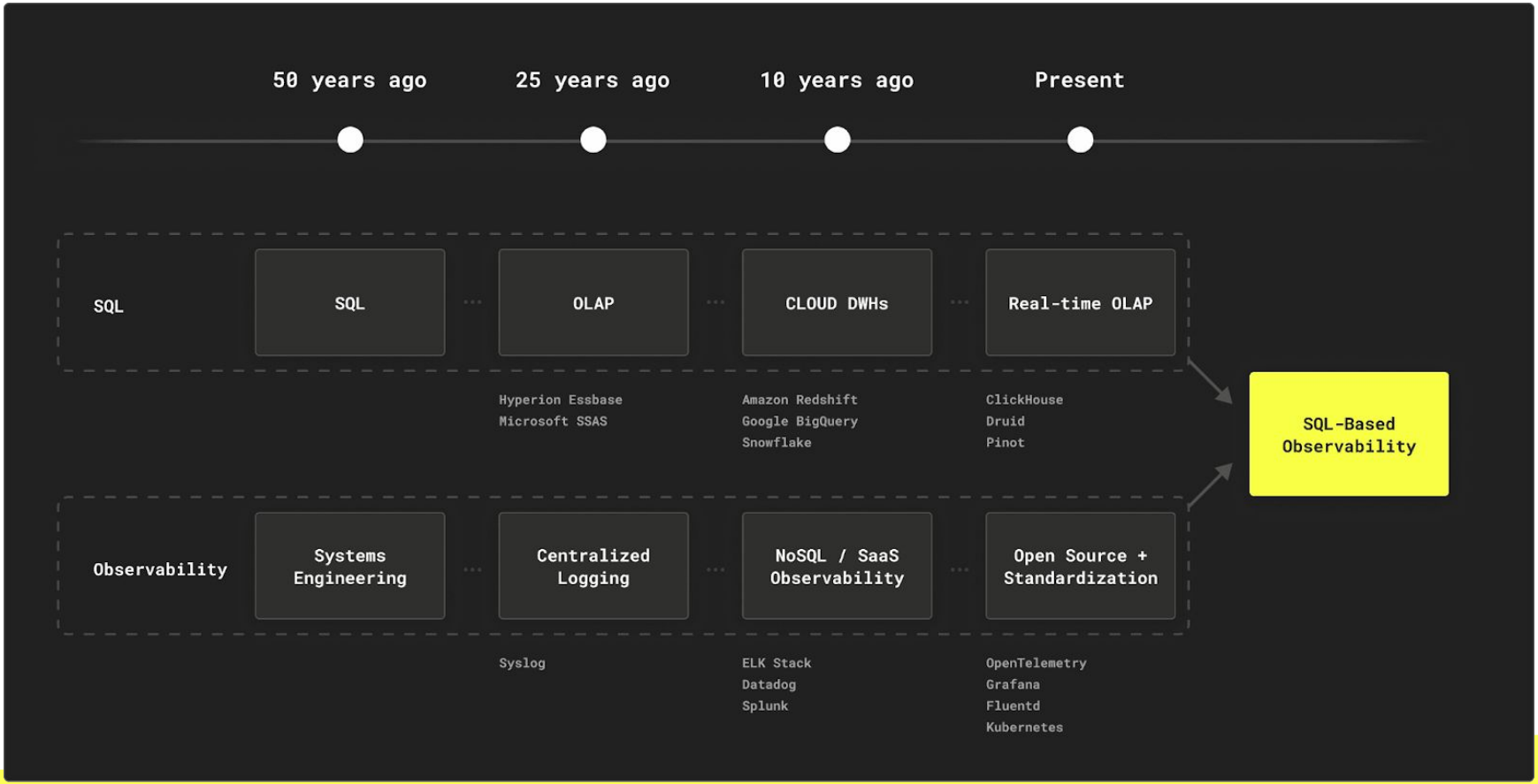
Evolution of real-time analytical databases



… at the same time,
practice of observability in DevOps matures



Until the two worlds start to collide



What is SQL-Based Observability?

“Observability is just another big data problem.”

BUT

Your choice of database matters in terms of performance and cost

AND

Significant advancements in databases built for DWH/Analytics can be applied to the observability space

Typical architecture for DIY Observability

Typical architecture for SQL-based Observability

Real-world examples

How others have done it, where to learn more

Adopting OLAP store for logging

Architecture

Log shippers + Kafka →
ClickHouse →
Custom connector to Kibana

The wins

Speed of ingestion, cost control

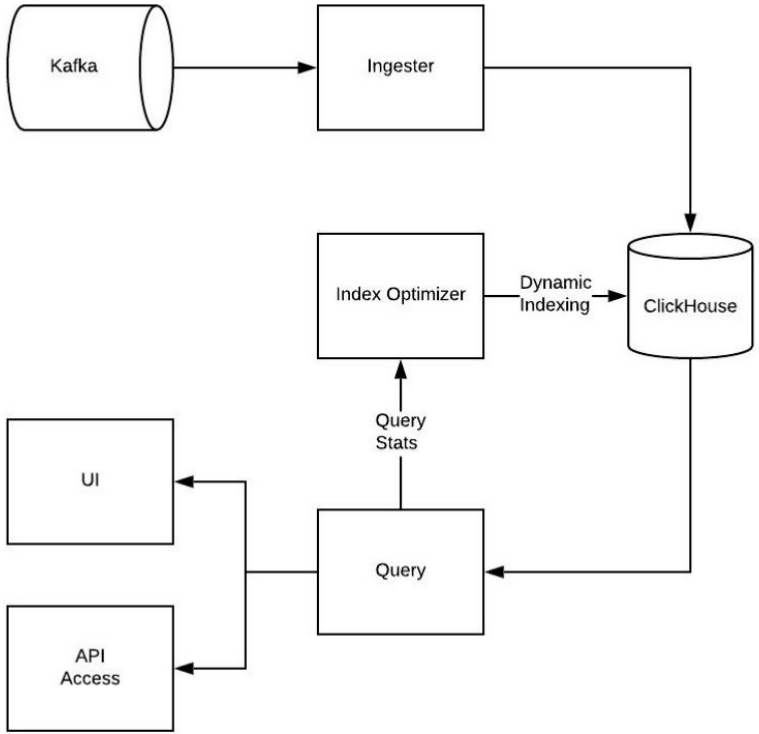
The tradeoffs

Stack administration, UI development

Uber

https://www.uber.com/blog/logging/
https://presentations.clickhouse.com/meetup40/uber.pdf

High-Level System Architecture



Uber

Adopting OLAP store for tracing

Architecture

OpenTelemetry →
ClickHouse →
Custom UI

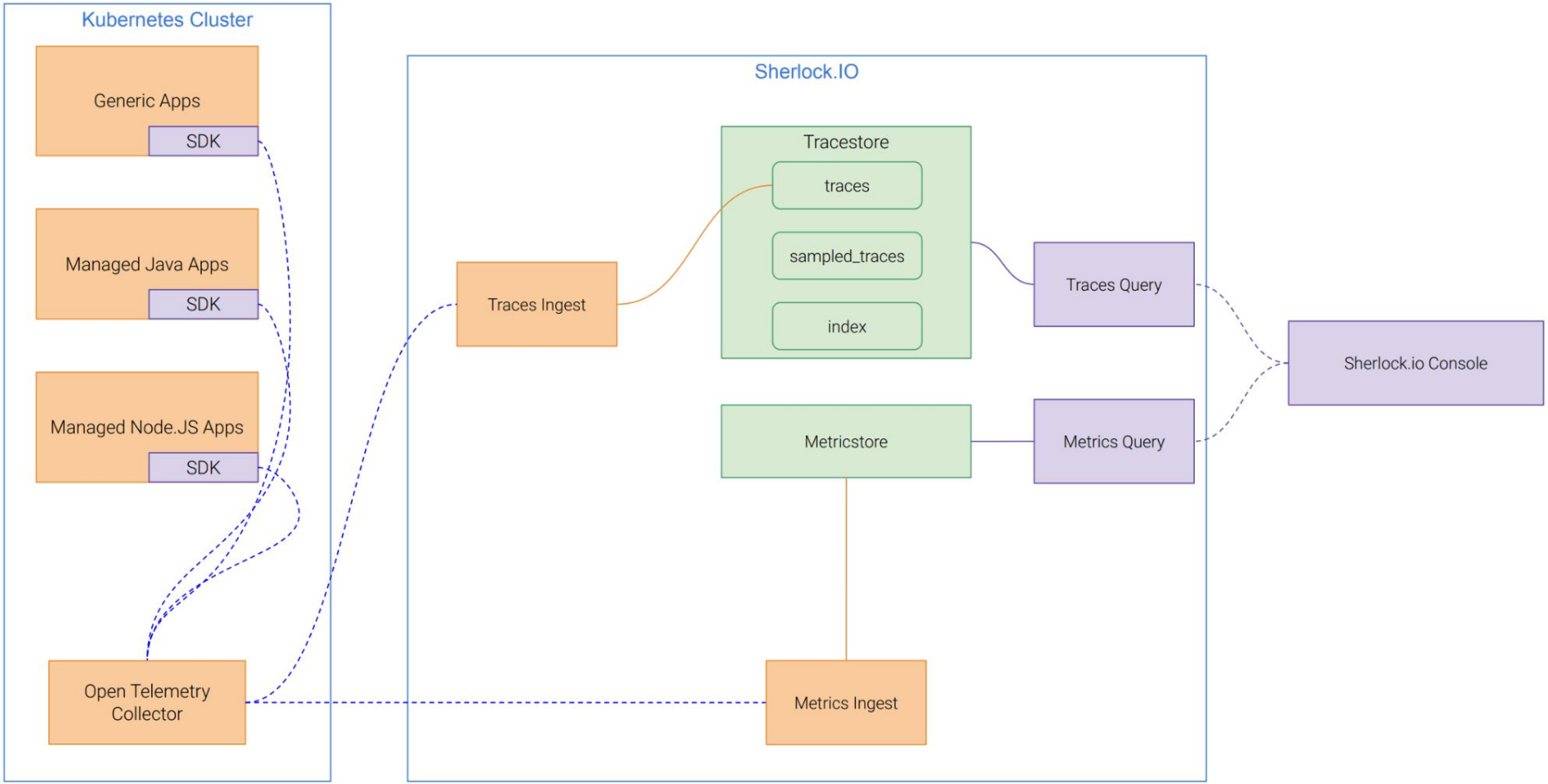
The wins

Data compression, open source licensing

The tradeoffs

Managing tiered OTel collectors

https://events.linuxfoundation.org/kubecon-cloudnativecon-europe/program/schedule/



Dogfooding ClickHouse across O11Y

Architecture

OpenTelemetry →
ClickHouse →
Grafana

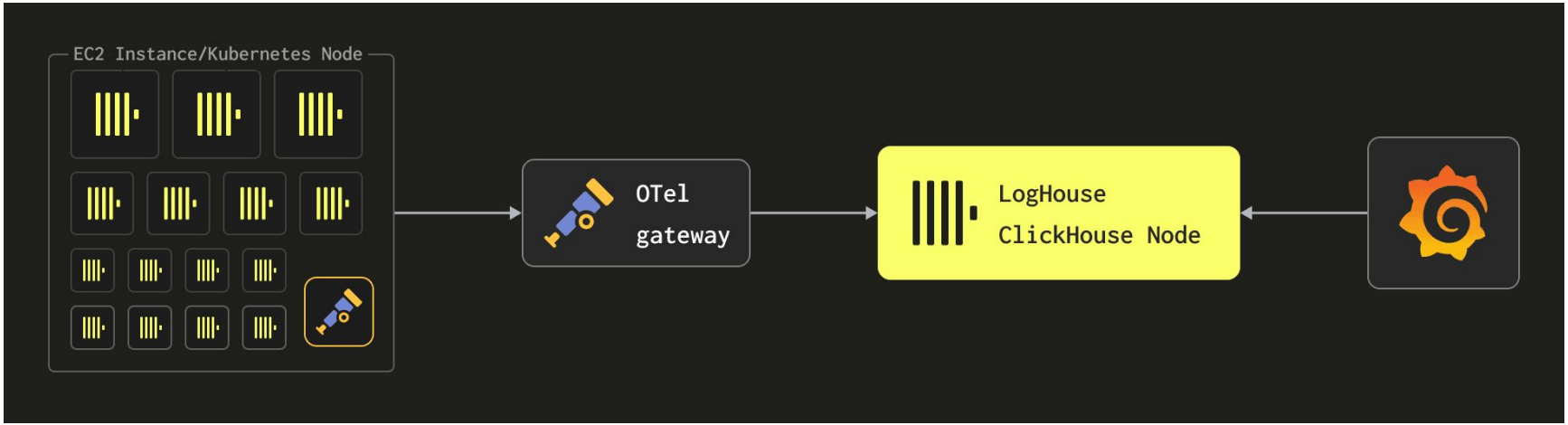
The wins

Granular log retention across infrastructure
Saved $26M/mo on Datadog? :-)

The tradeoffs

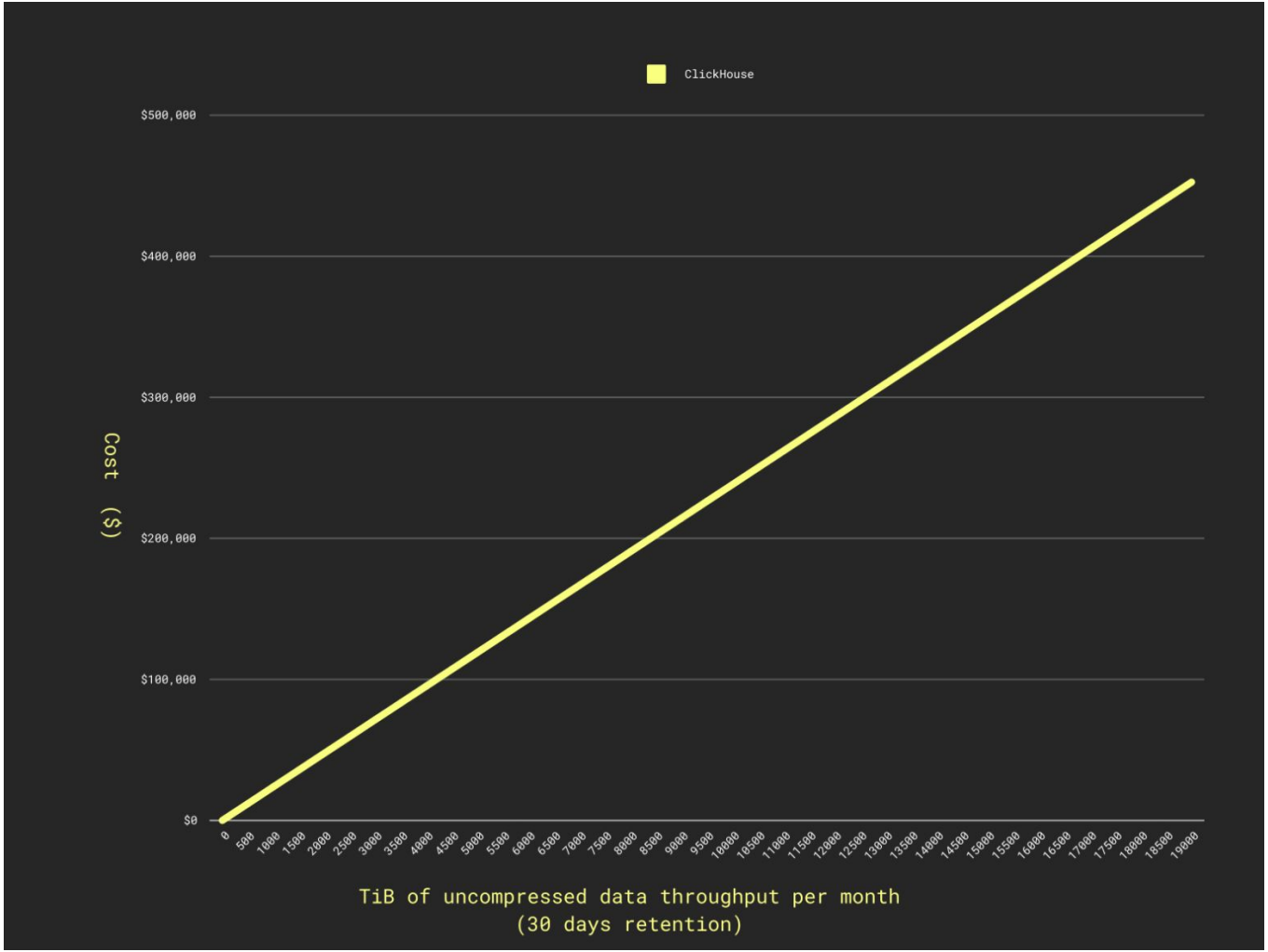
1.5 FTEs to build / maintain stack

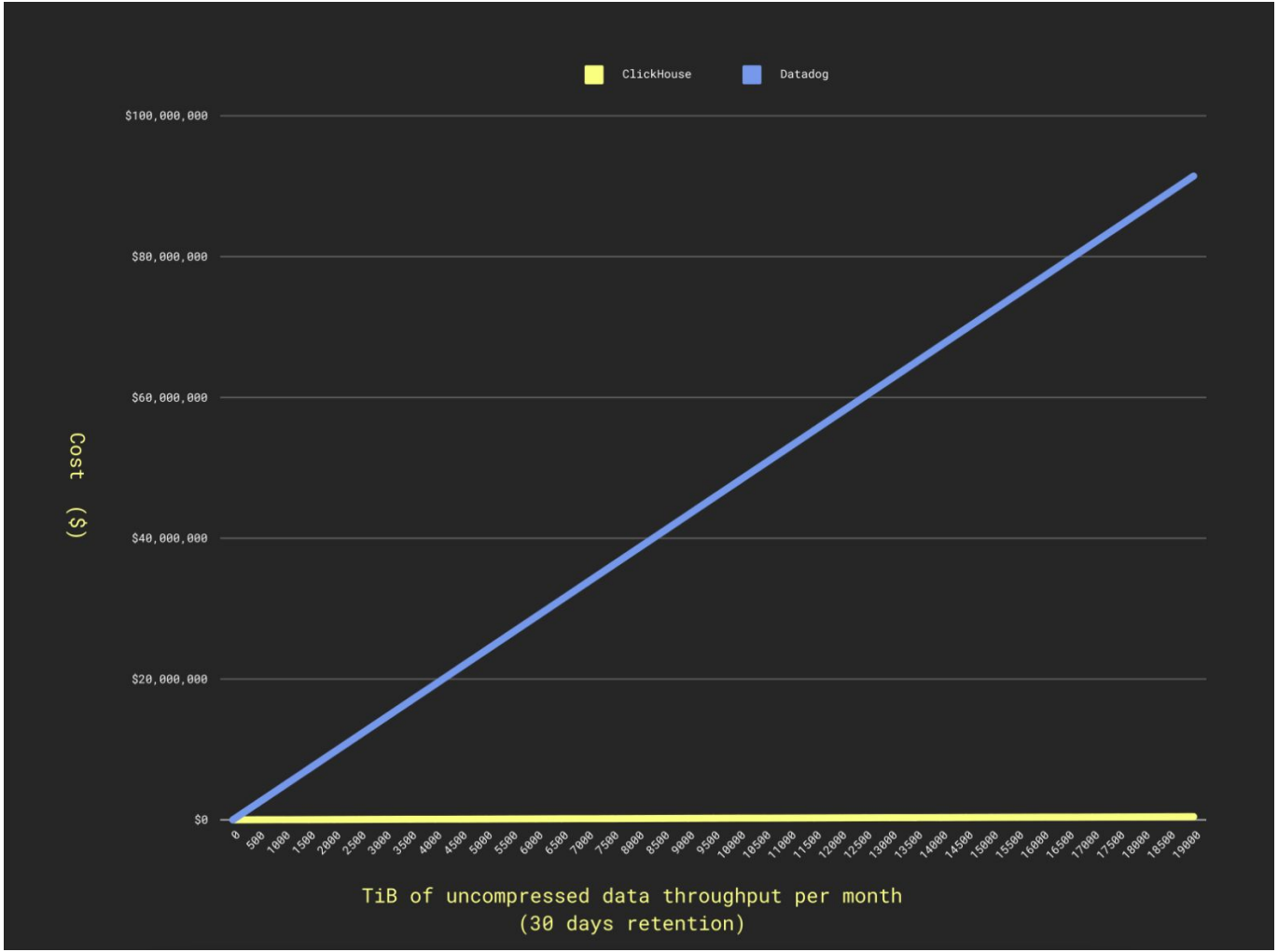
https://clickhouse.com/blog/building-a-logging-platform-with-clickhouse-and-saving-millions-over-datadog

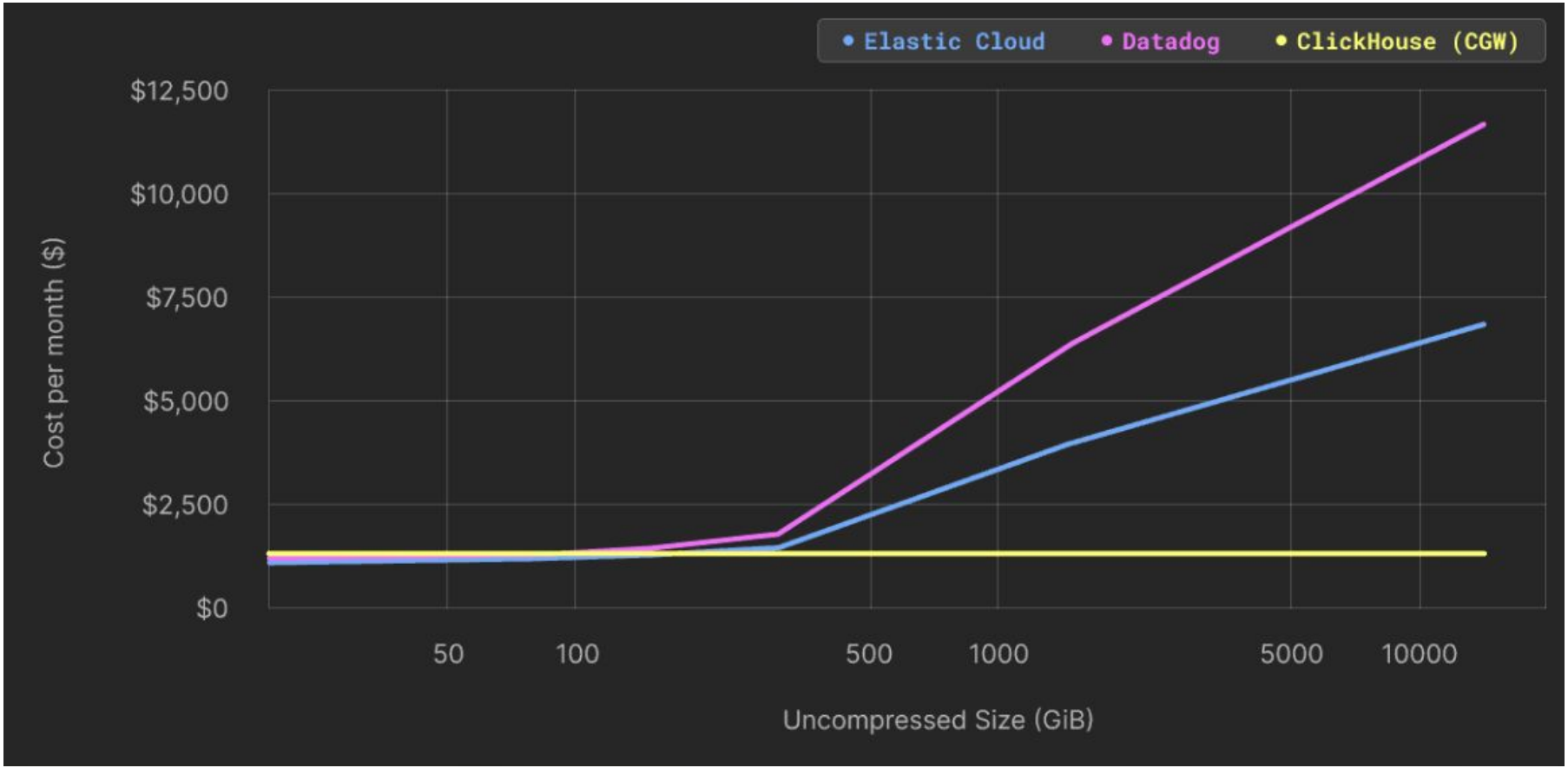


Total LogHouse Data

Uncompressed Size	Compressed Size	Ratio	Rows
19.11 PiB	1.13 PiB	17.0	36.96 trillion


ClickHouse
$500,000
$400,000
$300,000
$200,000
$100,000
$0
Cost ($)
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
6000
6500
7000
7500
8000
8500
9000
9500
10000
10500
11000
11500
12000
12500
13000
13500
14000
14500
15000
15500
16000
16500
17000
17500
18000
18500
19000
TiB of uncompressed data throughput per month
(30 days retention)


ClickHouse
Datadog
$100,000,000
$80,000,000
$60,000,000
$40,000,000
$20,000,000
$0
Cost ($)
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
6000
6500
7000
7500
8000
8500
9000
9500
10000
10500
11000
11500
12000
12500
13000
13500
14000
14500
15000
15500
16000
16500
17000
17500
18000
18500
19000
TiB of uncompressed data throughput per month
(30 days retention)


Elastic Cloud
Datadog
ClickHouse (CGW)
$12,500
$10,000
$7,500
$5,000
$2,500
$0
Cost per month ($)
50
100
500
1000
5000
10000
Uncompressed Size (GiB)

Common considerations

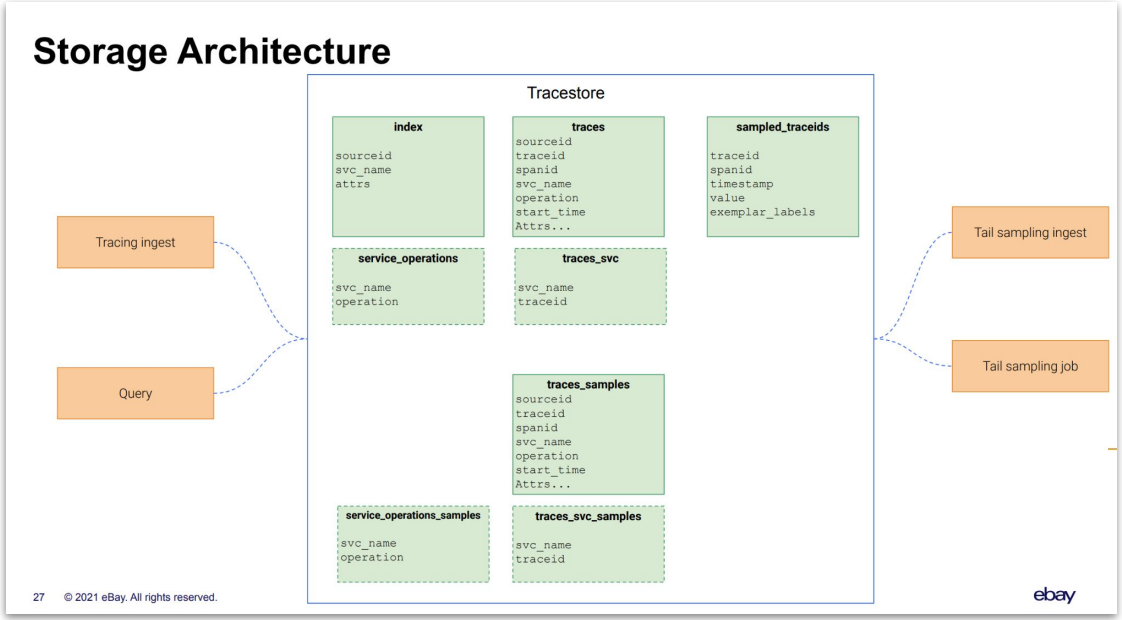
Key decisions & architecture patterns for SQL-Based Observability

ClickHouse

Schema considerations

Example: eBay

Even if you take inspiration from OTel, you will likely need to extend and tune your database schema



Schema considerations

Example: Uber

Complex data structures like Maps and Objects + ingest-level parsing for schema-agnostic platforms

Schema-agnostic data model

Our raw logs are formatted into JSON, whose schema can change gradually. While emitting log messages like “Job finished”, developers can tag them with key value pairs as context. The log message and tags are encoded in the output logs as fields. The tag values can be primitive types like number or string and composite types like array or object. In Uber, logs have 40+ fields on average, all treated equally by our platform and providing rich context.

To support schema evolution natively, we track all types seen in a field during ingestion in the log schema, as shown below. The schema is persisted and used during query execution, explained later. Each field type is tagged with a timestamp, which indicates when the type is observed and can be used to purge stale info from the schema.

<table>
<tr>
<td>

```
{
  "Msg": "Job finished",
  "User": "Foo"
}
```

</td>
<td>

```
{
  "Msg": "Job finished",
  "User": {
    "ID": 123,
    "Name": "Bar",
  }
}
```

</td>
<td>

```
{
  "Msg": "Job finished",
  "User": {
    "ID": "cx50yz",
    "Name": ["first","last"],
  }
}
```

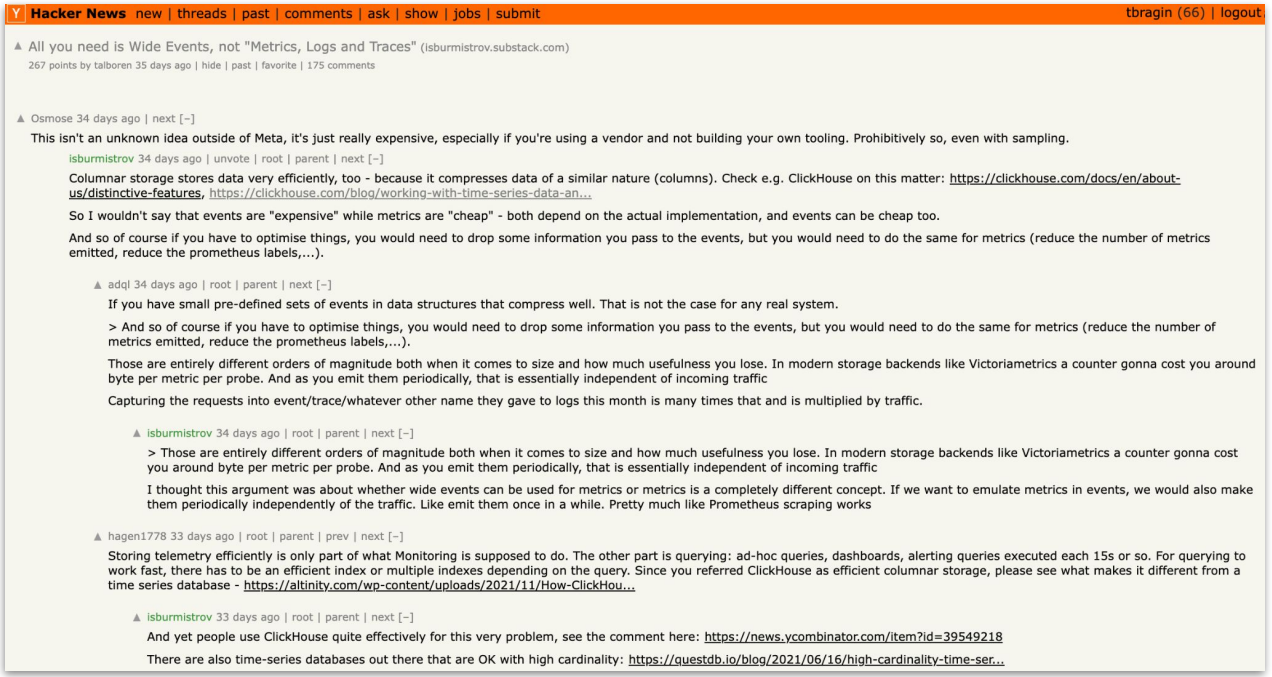
</td>
</tr>
<tr>
<td colspan="3">

```
// A field can have primitive values and JSON object values across logs.
"User": {
  String: 1596477600
}

// A field can have multiple types seen at different times.
"User.ID": {
  Number: 1596477601
  String: 1596477602
}

// Scalar and array type are separated.
"User.Name": {
  String: 1596477601
  StringArray: 1596477602
}

"Msg": {
  String: 1596477602
}
```

</td>
</tr>
</table>

Schema considerations

It may help to stop thinking about logs, metrics, traces separately and just think about “wide events”



Hacker News new | threads | past | comments | ask | show | jobs | submit — tbragin (66) | logout

▲ All you need is Wide Events, not "Metrics, Logs and Traces" (isburmistrov.substack.com)
267 points by talboren 35 days ago | hide | past | favorite | 175 comments

▲ Osmose 34 days ago | next [–]
This isn't an unknown idea outside of Meta, it's just really expensive, especially if you're using a vendor and not building your own tooling. Prohibitively so, even with sampling.

isburmistrov 34 days ago | unvote | root | parent | next [–]
Columnar storage stores data very efficiently, too - because it compresses data of a similar nature (columns). Check e.g. ClickHouse on this matter: https://clickhouse.com/docs/en/about-us/distinctive-features, https://clickhouse.com/blog/working-with-time-series-data-an...

So I wouldn't say that events are "expensive" while metrics are "cheap" - both depend on the actual implementation, and events can be cheap too.

And so of course if you have to optimise things, you would need to drop some information you pass to the events, but you would need to do the same for metrics (reduce the number of metrics emitted, reduce the prometheus labels,...).

▲ adql 34 days ago | root | parent | next [–]
If you have small pre-defined sets of events in data structures that compress well. That is not the case for any real system.

> And so of course if you have to optimise things, you would need to drop some information you pass to the events, but you would need to do the same for metrics (reduce the number of metrics emitted, reduce the prometheus labels,...).

Those are entirely different orders of magnitude both when it comes to size and how much usefulness you lose. In modern storage backends like Victoriametrics a counter gonna cost you around byte per metric per probe. And as you emit them periodically, that is essentially independent of incoming traffic

Capturing the requests into event/trace/whatever other name they gave to logs this month is many times that and is multiplied by traffic.

▲ isburmistrov 34 days ago | root | parent | next [–]
> Those are entirely different orders of magnitude both when it comes to size and how much usefulness you lose. In modern storage backends like Victoriametrics a counter gonna cost you around byte per metric per probe. And as you emit them periodically, that is essentially independent of incoming traffic

I thought this argument was about whether wide events can be used for metrics or metrics is a completely different concept. If we want to emulate metrics in events, we would also make them periodically independently of the traffic. Like emit them once in a while. Pretty much like Prometheus scraping works

▲ hagen1778 33 days ago | root | parent | prev | next [–]
Storing telemetry efficiently is only part of what Monitoring is supposed to do. The other part is querying: ad-hoc queries, dashboards, alerting queries executed each 15s or so. For querying to work fast, there has to be an efficient index or multiple indexes depending on the query. Since you referred ClickHouse as efficient columnar storage, please see what makes it different from a time series database - https://altinity.com/wp-content/uploads/2021/11/How-ClickHou...

▲ isburmistrov 33 days ago | root | parent | next [–]
And yet people use ClickHouse quite effectively for this very problem, see the comment here: https://news.ycombinator.com/item?id=39549218

There are also time-series databases out there that are OK with high cardinality: https://questdb.io/blog/2021/06/16/high-cardinality-time-ser...

https://isburmistrov.substack.com/p/all-you-need-is-wide-events-not-metrics
https://news.ycombinator.com/item?id=39529775

UI Considerations

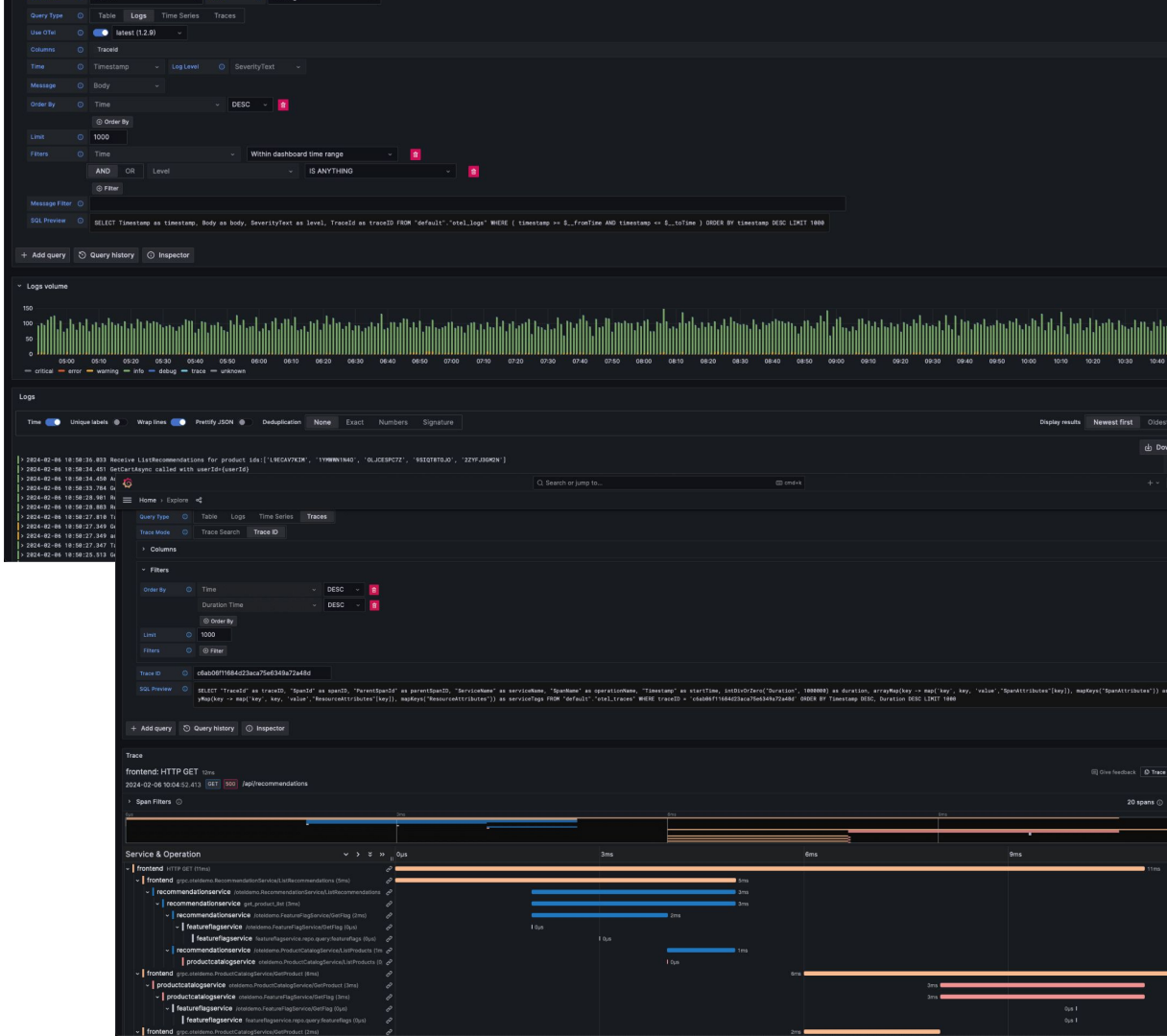
Grafana

Perses

Adapt Kibana (e.g. Quesma)
or Splunk (via DB Connect)

Build your own

Apache Superset



Query language considerations

“SQL is not compact enough compared to domain-specific query languages”

A simple query ?

```
source=events level=”warning”
| STATS avg(duration) BY level
| FIELDS level, avg(duration) AS avg_dur
| sort - avg_dur | head 10
```

```
GET events/_search
{
  "size": 0,
  "_source": false,
  "track_total_hits": -1,
  "aggregations": {
    "groupby": {
      "composite": {
        "size": 10,
        "sources": [
          {
            "4e8796da": {
              "terms": {
                "field": "level.keyword",
                "missing_bucket": true,
                "order": "asc"
              }
            }
          }
        ]
      },
      "aggregations": {
        "c3318afb": {
          "avg": {
            "field": "duration"
          }
        }
      }
    }
  }
}
```

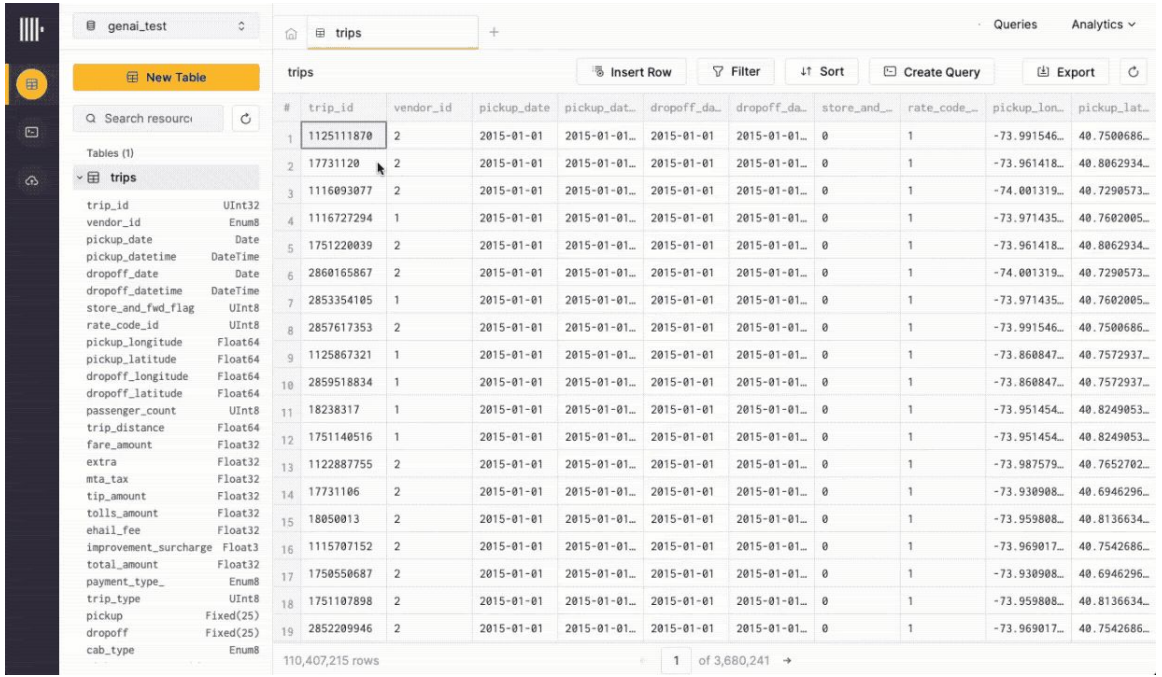
In good old SQL …

```
SELECT
    level,
    avg(duration) AS dur
FROM events
GROUP BY level
ORDER BY dur DESC
```

Query language considerations

“DevOps engineers do not want to write SQL queries”

Valid question, but with copilots being the norm, do they still matter?



Multi-tenancy considerations

Example: Uber

Consider datastore ability to limit resources by table, user, session

Unified Multi-Tenant Storage Platform

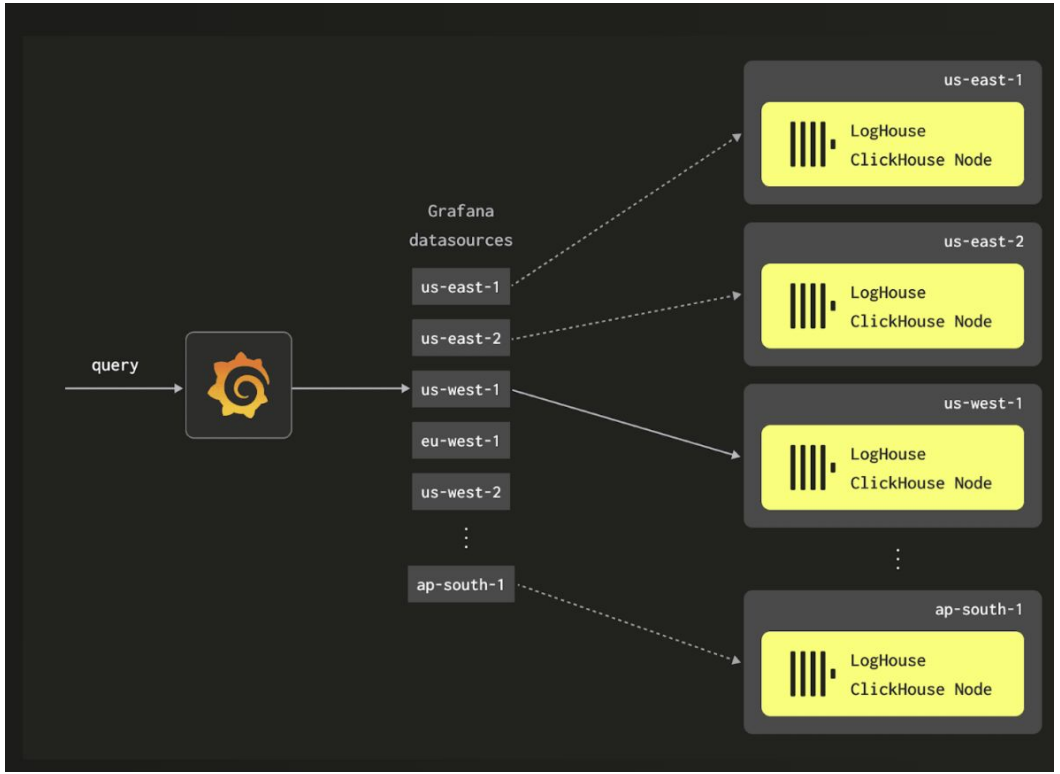
- ClickHouse natively supports zero lock contention among concurrent reads and writes
- Service placement: single-tenant vs multi-tenant
  - Isolate heavy log producers, heavy log consumers
  - Co-locate everything else
  - Limit the impact of co-location, add service in order-by
- Workload isolation
  - Configure query parallelism per query
  - Eventually limit total query resource usage per node
  - Query cost accounting, defense against expensive queries

Multi-region considerations

Example: ClickHouse

Per region data collection / storage, cross-region queries

Resilient to AZ outage but not region outage



SQL OLAP datastore considerations

- “Real-time”
  - fast ingestion (inc. ability to land, then transform, if needed)
  - common queries in small # of seconds (on large amounts of data)
- Data compression
  - 10-100x compression of common observability datasets
- Separation of storage and compute
  - Support for large dataset storage on object storage, scale compute separately
- Interoperability
  - Support for common O11Y shippers (OTel) and UI (Grafana)
- SQL compliance
  - How close to ANSI SQL compliant is it? (Every SQL flavor is a bit different)
- TCO
  - Disruptive TCO benefits - ~10x to account for switching costs

Choice of SQL OLAP datastore matters

	ClickHouse	Apache Druid	Apache Pinot	BigQuery
Real-time ingest & queries	✔ Best	✔ Ok	✔ Ok	✘ Poor
Compression	✔ Best	✔ Ok	✔ Ok	✔ Better
Sep storage & compute	✔	✘	✘	✔
Interoperability - **OTel** - **Grafana**	✔ ✔	✘ ✔ (no logging & tracing support)	✘ ✘	✘ ✔ (no logging & tracing support)
SQL compliance	✔ Good	✘ Poor	✘ Poor	✔ Best
TCO improvements	✔ 5-10x	1-2x	1-2x	depends on how often you query

A look into the future

Where this use case is headed?

My prediction for CY 2024 - 2026

- Observability costs will continue to gain scrutiny within organizations due to pressure on COGS and internal cost centers
- DevOps engineers will search for disruptive solutions to these challenges
- Real-time analytical (or OLAP) datastores that provide 10x benefits will continue to gain prominence in observability use cases

Further reading

https://clickhouse.com/blog/the-state-of-sql-based-observability

Uber logging blog:
https://www.uber.com/blog/logging/

eBay tracing talk:
https://events.linuxfoundation.org/kubecon-cloudnativecon-europe/program/schedule/

LogHouse blog:
https://clickhouse.com/blog/building-a-logging-platform-with-clickhouse-and-saving-millions-over-datadog

Find me

https://x.com/tbragin

https://www.linkedin.com/in/tbragin/

https://medium.com/@tbragin