PostHog ♥ ClickHouse

How we turned this quaint place into our event mansion

Jams Greenhill
@fuziontech

Eric Duong

What is PostHog?

Open source product analytics suite

Event pipelines

Analytics

Session recordings

Feature flags

Heatmaps

Export to data warehouse

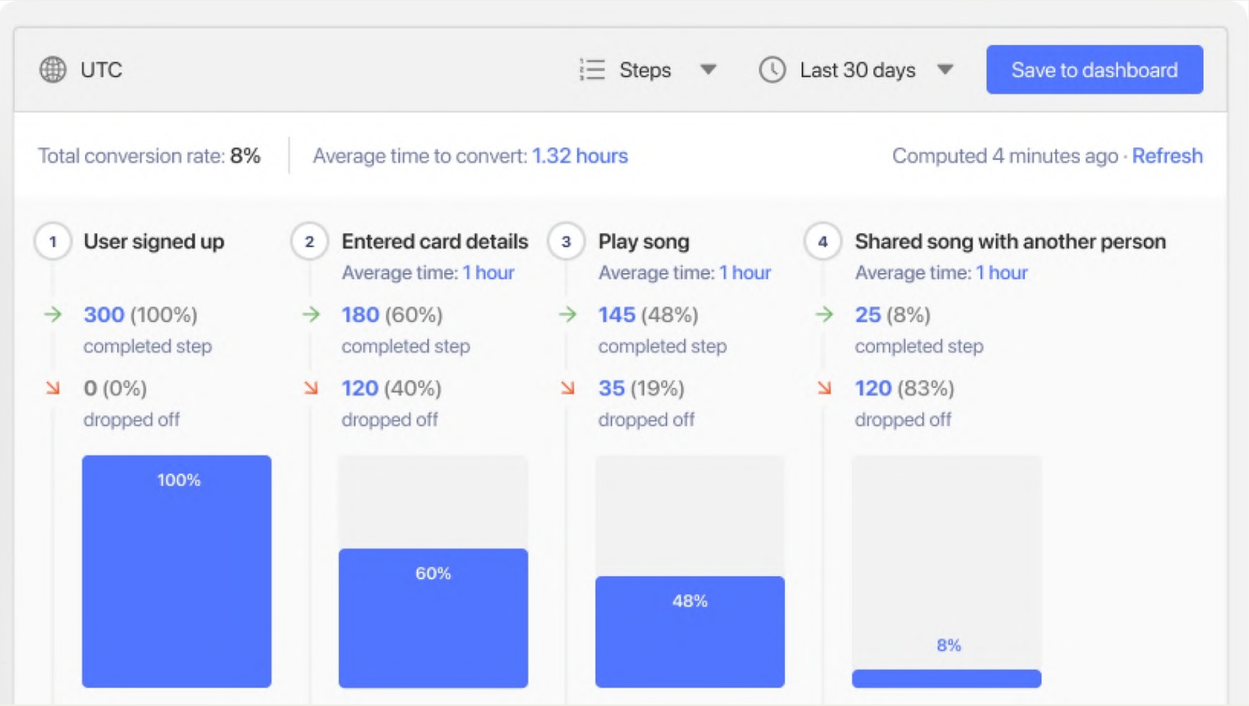
- Consuming events since January 2020
- Completely distributed

PostHog

Defining characteristics

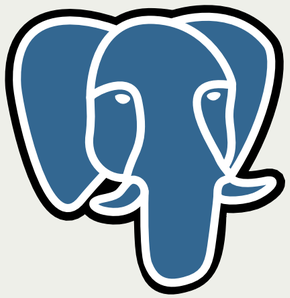
A data warehouse in a box

- As close to real time as possible
- Continually growing feature set
- Accept and filter on arbitrary user-defined properties in event payloads
- You can host and own your data yourself!



PostHog

Version 0.5: Postgres



MVP

- All queries written in django ORM
- PostHog features require relations, perfect for Postgres
- Postgres is well supported by the community and easy to deploy
- We pushed Heroku Postgres to its absolute limit. Had to move to something more capable!

How we evaluated ClickHouse

OLAP databases we thought about using:

- **Pinot**
- **Presto**
- **Druid**
- **TimescaleDB**
- **CitusDB**
- **ClickHouse**

How we evaluated ClickHouse

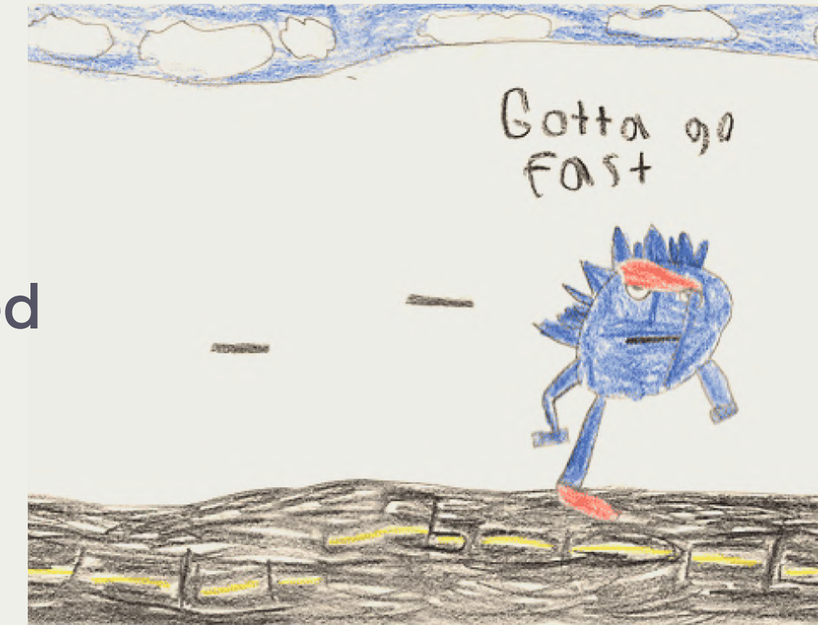
Basically came down to three factors:

- Speed
- Complexity of management
- Query interface

Why ClickHouse

How we made the decision

- Reviewed a ton of benchmarks
- Looked at who is using what and asked how they liked it
  - Sentry - ClickHouse for pretty much everything
  - Cloudflare - 6m requests per second
  - Uber - Logs
  - Yandex
- Setup a test cluster and was blown away by the speed and ease of setup



PostHog

What is a ClickHouse?

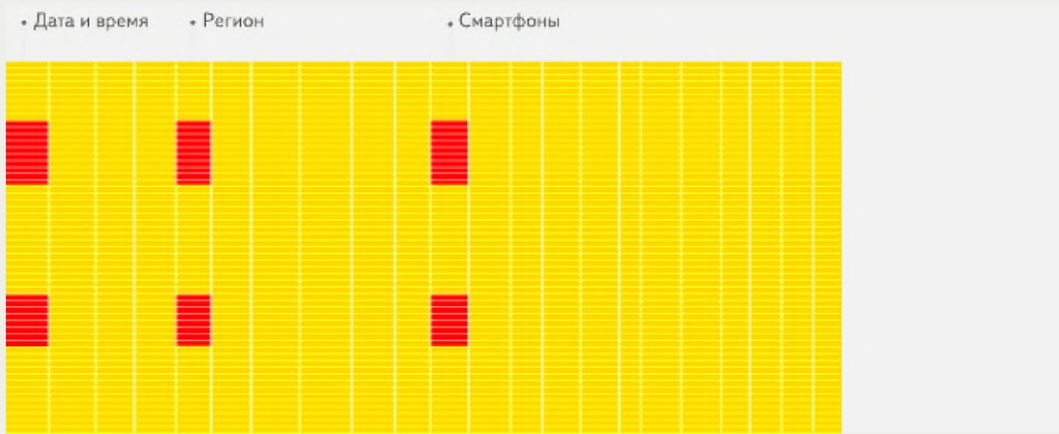
How we made the decision

- ClickHouse powers Yandex Metrica (Russian Google Analytics)
  - 20 billion events a day. Over 13 trillion records in total. Reports are built on the fly from **non-aggregated** data.
- Column-oriented database management system (DBMS) for online analytical processing (OLAP) of queries.
- Written in C++
- Tightly coupled compute and storage

PostHog

What makes ClickHouse special

- Excellent compression
- Process data from disk (cheap)
- Vector computation
- Near real-time data updates (based on sort keys)
- Sorted data allows primary and secondary indexes
- Data skipping indexes
- No query planner
- Support for approximated calculations
- Easy replication & sharding (more on that later)



PostHog

What makes ClickHouse painful

- Mutations are not really supported and very expensive on disk throughput
- Transactions are not supported
- Sparse index is not a real index. Does not help with grabbing a single row by key
- Resharding is painfully manual using ClickHouse-Copier

What is an EventMansion

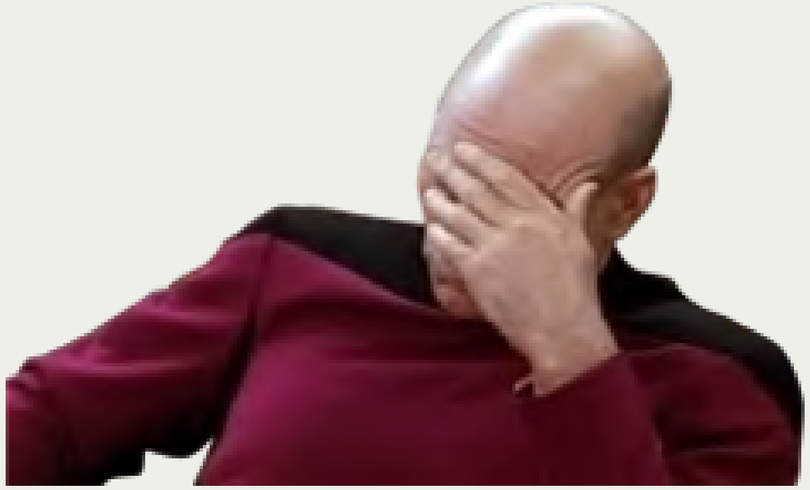
- *Users should get as close to real time results as possible*
  - **ClickHouse performance along with some clever architecting allows us to continue providing users with near real time results**

- *There's a continually growing featureset*
  - **ClickHouse SQL has been developed to resemble popular syntaxes such as those from Postgres and MySQL (joins, window functions, etc.)**

What is an EventMansion

- *The system should be able to accept and filter on user-defined properties in event payloads*
  - **Features such as materialized columns allowed us to tailor architecture to perform well for our use cases**
- *The system should be as easily deployable as possible to cater to users who want to own their data*
  - **Integrated solutions such as kafka engines helped us get off the ground quickly**

Initial learnings



- We tried to extract JSON key, values
  - Arrays can be slow [k1, k2, k3, ...], [v1, v2, v3, ...]
  - KV lookup table even slower
  - Leverage columnar advantages as much as you can!
- JSONExtractString is surprisingly fast, but still expensive for CPU on large JSON objects
- Developed our own way to serialize DOM hierarchies to make querying the DOM very efficient
  - Thanks google/re2!
- Don’t mutate if you can help it!

Making the switch

- We made the switch by deploying ClickHouse and ingesting events to both our ClickHouse and Postgres database, effectively duplicating our ingestion
- Reimplemented our analytic queries one by one using feature flags
- Once everything was migrated, we simply stopped storing into Postgres

How we deploy ClickHouse

- On custom i3en.12xlarge instances with local NVMe storage
  - We use CloudFormation + Ansible to configure these.
- Using ClickHouse-Operator (made by Altinity!)
  - Customer deployed PostHog is all managed through our Helm Chart. We did this because of the amazing ClickHouse-Operator offered by Altinity that makes our job much easier.

ClickHouse on app.posthog.com

- 4 x i3en.12xlarge
  - 48 vCPU
  - 384 GB Memory
  - 4 x 7,500 gb NVMe SSD
    - Raid 1+0
  - 50 Gbps Network Bandwidth
- 2 x t3a.2xlarge
  - 8 vCPU
  - 32 GB Memory
  - 8 TB EBS GP3
  - Up to 5 Gbps Network Bandwidth
- Total
  - 208 vCPU
  - 1,600 GB Memory
  - 30 TB NVMe SSD
  - 16 TB GP3 EBS

PostHog

Storage Optimized

Storage optimized instances are designed for workloads that require high, sequential read and write access to very large data sets on local storage. They are optimized to deliver tens of thousands of low-latency, random I/O operations per second (IOPS) to applications.

I3 | I3en | D2 | D3 | D3en | H1

This instance family provides dense Non-Volatile Memory Express (NVMe) SSD instance storage optimized for low latency, high random I/O performance, high sequential disk throughput, and offers the lowest price per GB of SSD instance storage on Amazon EC2. I3en also offers Bare Metal instances (i3en.metal), powered by the Nitro System, for non-virtualized workloads, workloads that benefit from access to physical resources, or workloads that may have license restrictions.

Features:

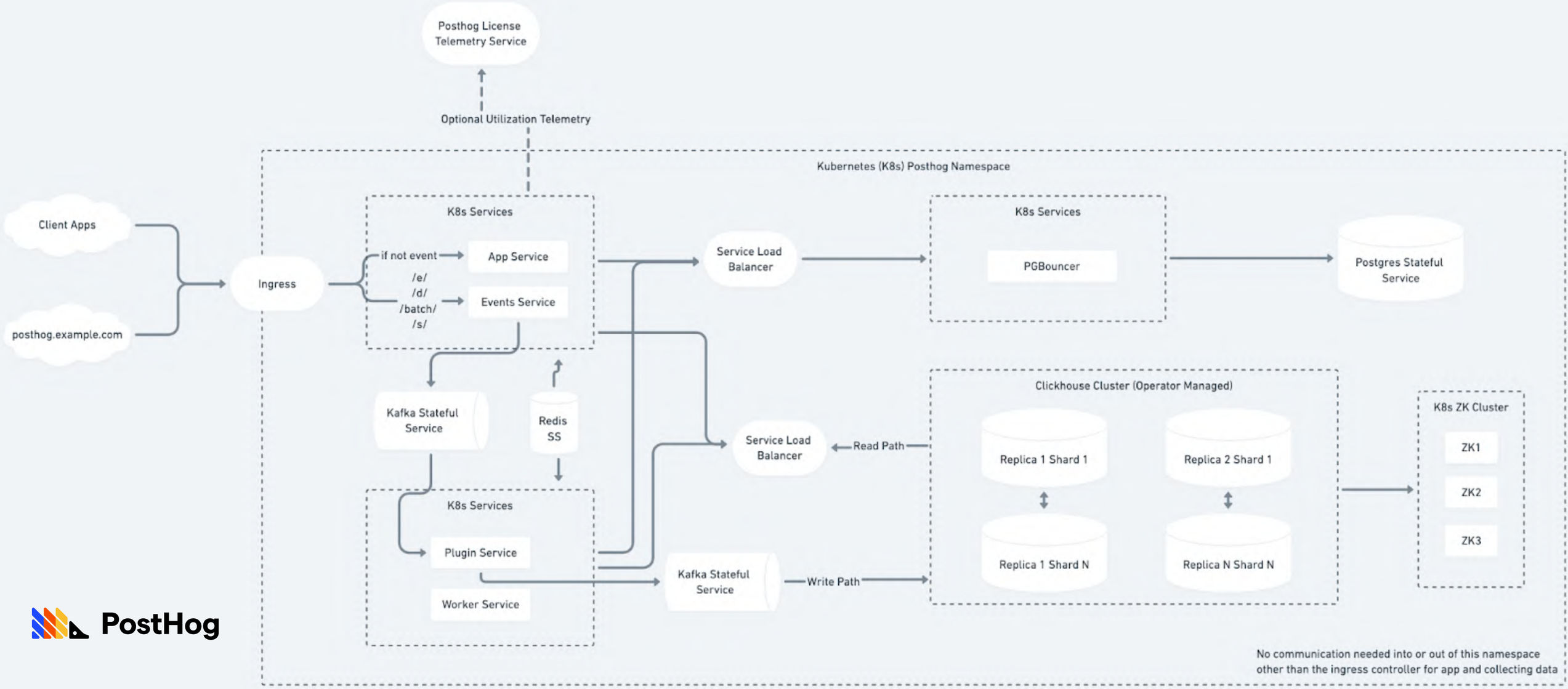
- Up to 60 TB of NVMe SSD instance storage
- Up to 100 Gbps of network bandwidth using Elastic Network Adapter (ENA)-based Enhanced Networking
- High random I/O performance and high sequential disk throughput
- Up to 3.1 GHz Intel® Xeon® Scalable (Skylake) processors with new Intel Advanced Vector Extension (AVX-512) instruction set
- Powered by the AWS Nitro System, a combination of dedicated hardware and lightweight hypervisor
- Support bare metal instance size for workloads that benefit from direct access to physical processor and memory
- Support for Elastic Fabric Adapter on i3en.24xlarge

Instance	vCPU	Mem (GiB)	Local Storage (GB)	Network Bandwidth
i3en.large	2	16	1 x 1,250 NVMe SSD	Up to 25 Gbps
i3en.xlarge	4	32	1 x 2,500 NVMe SSD	Up to 25 Gbps
i3en.2xlarge	8	64	2 x 2,500 NVMe SSD	Up to 25 Gbps
i3en.3xlarge	12	96	1 x 7,500 NVMe SSD	Up to 25 Gbps
i3en.6xlarge	24	192	2 x 7,500 NVMe SSD	25 Gbps
i3en.12xlarge	48	384	4 x 7,500 NVMe SSD	50 Gbps
i3en.24xlarge	96	768	8 x 7,500 NVMe SSD	100 Gbps
i3en.metal	96	768	8 x 7,500 NVMe SSD	100 Gbps

All instances have the following specs:

- 3.1 GHz all core turbo Intel® Xeon® Scalable (Skylake) processors
- Intel AVX†, Intel AVX2†, Intel AVX-512†, Intel Turbo
- EBS Optimized
- Enhanced Networking

Helm architecture



PostHog

ClickHouse table engines

ClickHouse allows you to set an engine when creating data tables depending on the data's characteristics

- ReplacingMergeTree - Dedupes for us!
- CollapsingMergeTree - Mutations, but without actual mutations
- VersionedCollapsingMergeTree ^ But better
- Kafka - Read from Kafka topic as if it were local storage
- Distributed - Scale out
- MaterializedView - View, but persisted to disk
- S3 - Cold Storage

PostHog

To ORM or not to ORM

(Object-relational mapping)

- Previously relied on django ORM to query data
- Given the complexity of our queries going forward, we chose not to use any of the existing clickhouse ORMs
  - More control
  - Easier to debug
  - Less interfaces to manage

Query characteristics

- ClickHouse syntax is familiar (datepart() vs toStartOfMonth())
- Not all expected features exist yet (CTEs were limited, window functions just)
- But the ones that do are powerful
  - Window functions came in handy for rewriting funnel queries 🌶️
  - ClickHouse has doubled down in providing extra functionality for array processing (Example: arraySplit to discover session boundaries instead of neighbor functions)
- Joins help us continue manipulating relational data

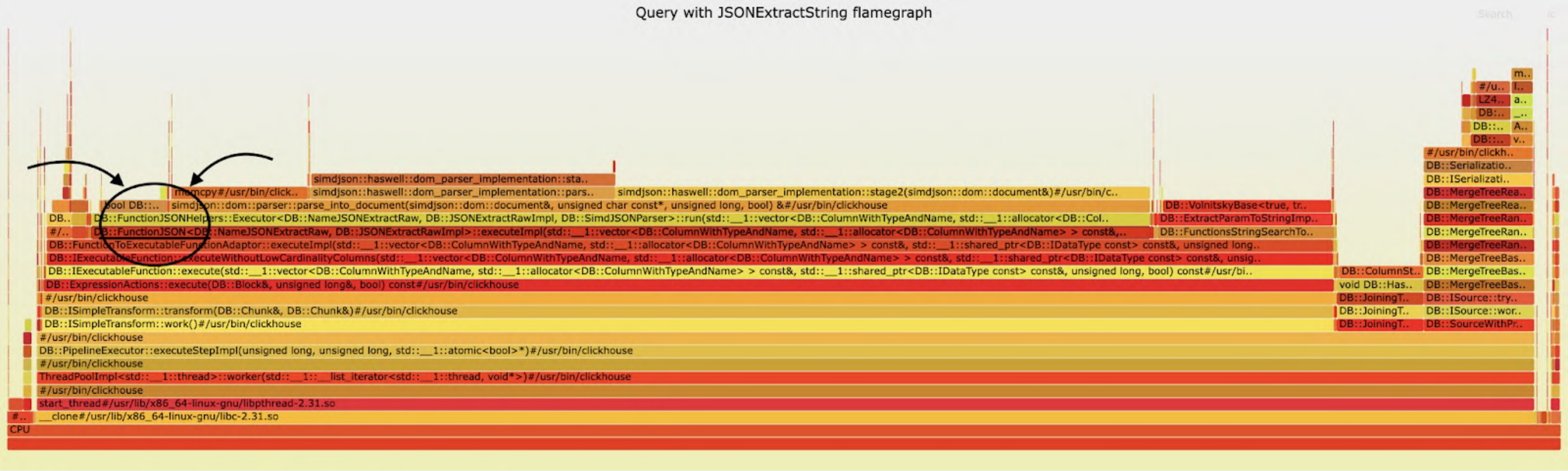
Testing

Keeping the ship sailing

- Integration tests is our main focus due to variability in queries
- Recently added snapshot tests to ensure specific query shapes
- Airspeed Velocity (ASV) is used to run benchmark tests on a fresh deployment of posthog loaded with a sample set of data

Latest learnings

Queries against event properties was slow



PostHog

Solution:

Materialized columns!

- 3.4x improvement in query time
- 81% reduction in bytes read from disk

```
ALTER TABLE events
ADD COLUMN mat_$current_url
VARCHAR MATERIALIZED JSONExtractString(properties_json, '$current_url')
```

PostHog

The future

- Build functions at the source
- Materialize/precalculate more data (actions)
- MaterializedPostgreSQL engine experiments
- Leverage Altinity-hosted ClickHouse

PostHog

Questions?!

PostHog