Building an Observability platform with ClickHouse

Ankit Nayan (CTO, SigNoz)

https://github.com/SigNoz/signoz

About Me

- Building SigNoz, part of YCombinator
- 8 yrs experience in build scalable tech architecture
- Passionate about distributed systems
- Love Himalayas and trekking

SigNoz - Open source observability platform

👉 Metrics + Traces in a single pane

👉 Powerful trace filtering and aggregation capabilities

👉 Out-of-box like SaaS solutions, requiring minimal dev effort

👉 Native backend and frontend to OpenTelemetry

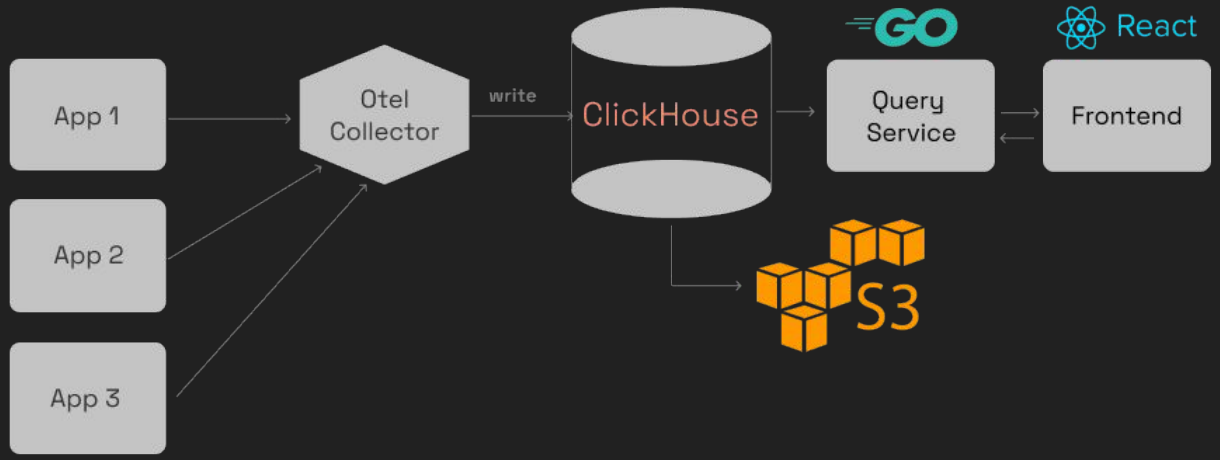
👉 Can be run within your own cloud

https://github.com/SigNoz/signoz

Quick Demo

https://github.com/SigNoz/signoz

Architecture of SigNoz



Components

- OpenTelemetry Collector
- ClickHouse
- Query Service
- Frontend

https://github.com/SigNoz/signoz

Why ClickHouse for storing Observability data

- Wide tables with each query processes large number of rows but a few columns
- Running aggregate queries
- Distributed columnar datastore and blazing fast queries
- Compression capabilities
- Hot and cold storage

Tracing data model in ClickHouse

Using `Array(String)` to store tagKeys and tagValues

Extracting frequently used tags to columns

https://github.com/jaegertracing/jaeger/issues/1438

```
CREATE TABLE IF NOT EXISTS signoz_index (
  timestamp DateTime64(9) CODEC(Delta, ZSTD(1)),
  traceID String CODEC(ZSTD(1)),
  spanID String CODEC(ZSTD(1)),
  parentSpanID String CODEC(ZSTD(1)),
  serviceName LowCardinality(String) CODEC(ZSTD(1)),
  name LowCardinality(String) CODEC(ZSTD(1)),
  kind Int32 CODEC(ZSTD(1)),
  durationNano UInt64 CODEC(ZSTD(1)),
  tags Array(String) CODEC(ZSTD(1)),
  tagsKeys Array(String) CODEC(ZSTD(1)),
  tagsValues Array(String) CODEC(ZSTD(1)),
  statusCode Int64 CODEC(ZSTD(1)),
  references String CODEC(ZSTD(1)),
  externalHttpMethod Nullable(String) CODEC(ZSTD(1)),
  externalHttpUrl Nullable(String) CODEC(ZSTD(1)),
  component Nullable(String) CODEC(ZSTD(1)),
  dbSystem Nullable(String) CODEC(ZSTD(1)),
  dbName Nullable(String) CODEC(ZSTD(1)),
  dbOperation Nullable(String) CODEC(ZSTD(1)),
  peerService Nullable(String) CODEC(ZSTD(1)),
  INDEX idx_tagsKeys tagsKeys TYPE bloom_filter(0.01) GRANULARITY 64,
  INDEX idx_tagsValues tagsValues TYPE bloom_filter(0.01) GRANULARITY 64,
  INDEX idx_duration durationNano TYPE minmax GRANULARITY 1
) ENGINE MergeTree()
PARTITION BY toDate(timestamp)
ORDER BY (serviceName, -toUnixTimestamp(timestamp))
```

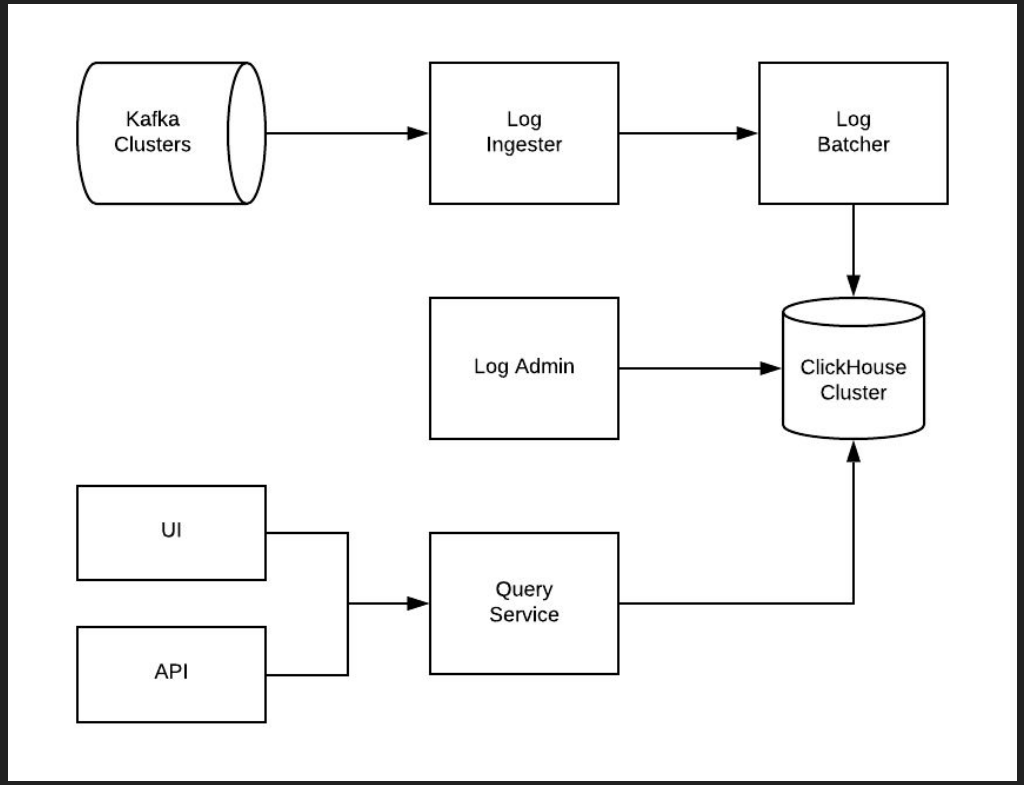
Storage Performance of Clickhouse

```
SELECT
    sum(marks) AS marks,
    sum(rows) AS rows,
    formatReadableSize(sum(data_compressed_bytes)) AS compressed,
    formatReadableSize(sum(data_uncompressed_bytes)) AS uncompressed,
    toDecimal64(sum(data_uncompressed_bytes) / sum(data_compressed_bytes), 2) AS compression_ratio,
    formatReadableSize(sum(data_compressed_bytes) / rows) AS bytes_per_row,
    formatReadableSize(sum(primary_key_bytes_in_memory)) AS pk_in_memory
FROM system.parts
```

marks	rows	compressed	uncompressed	compression_ratio	bytes_per_row	pk_in_memory
1363	8833744	307.58 MiB	3.10 GiB	10.32	36.51 B	15.40 KiB

https://github.com/SigNoz/signoz

ClickHouse for Logs Management at Uber



[1] https://eng.uber.com/logging/

Comparison with Elastic

- A single ClickHouse node could ingest 300K logs per second, 10x a single ES node
- >80% queries are aggregation queries, such as terms, histogram and percentile aggregations. Elastic is not designed to support fast aggregation

```
CREATE TABLE <table_name>
(
      // Common metadata fields.
      _namespace                String,
      _timestamp                 Int64,
      hostname                  String,
      zone                      String,
      ...

      // Raw log event.
      _source                   String,

      // Type-specific field names and field values.
      string.names              Array(String),
      string.values             Array(String),
      number.names              Array(String),
      number.values             Array(Float64),
      bool.names                Array(String),
      bool.values               Array(UInt8),

      // Materialized fields
      bar.String                String,
      foo.Number                Float64,
      ...
)
...
```

[1] https://eng.uber.com/logging/

Metrics Data model for ClickHouse

```
CREATE TABLE time_series (
    date Date CODEC(Delta),
    fingerprint UInt64,
    labels String
)
ENGINE = ReplacingMergeTree
    PARTITION BY date
    ORDER BY fingerprint;

CREATE TABLE samples (
    fingerprint UInt64,
    timestamp_ms Int64 CODEC(Delta),
    value Float64 CODEC(Delta)
)
ENGINE = MergeTree
    PARTITION BY toDate(timestamp_ms / 1000)
    ORDER BY (fingerprint, timestamp_ms);
```

{"__name__":"up","instance":"clickhouse_exporter_1:9116","job":"clickhouse"}

Metrics @ SigNoz

- Metrics with ClickHouse uses 1.16 byte vs 1.37 byte with Prom local storage

marks	rows	compressed	uncompressed	compression_ratio	bytes_per_row	pk_in_memory
1073358	8786279396	9.49 GiB	236.34 GiB	24.89	1.16 B	16.31 MiB

- PromQL like query model for reading metrics data from ClickHouse

https://github.com/SigNoz/signoz

Planned Improvements

- Another table to store complete span and index by traceID
- Use `ReplicatedMergeTree` for data redundancy
- Use `SummingMergeTree` for faster aggregation queries
- Use distributed tables for distributed queries
- Separation of roles using separate data and query nodes leads to independent scaling and independent hardware SKUs

https://github.com/SigNoz/signoz

Summary

- ClickHouse seems a good fit for Observability data
- Uber has used ClickHouse at scale for logs
- Aggregation of span data more efficient with OLAP DBs
- Less resource intensive for smaller scale compared to Druid, ideal for open source project like ours

https://github.com/SigNoz/signoz

Thank You

Give SigNoz a try. Check out our Github repo at

https://github.com/SigNoz/signoz


ankit@signoz.io


@ankitnayan