

ClickHouse® & OpenTelemetry

Tracing a Cloud-Native Database + Storing Traces with ClickHouse

Altinity® is a Registered Trademark of Altinity, Inc. ClickHouse® is a registered trademark of ClickHouse, Inc.; Altinity is not affiliated with or associated with ClickHouse, Inc.



Josh Lee
Open Source Advocate



Maciej Bak
Support Engineer



ALTINITY® Build better on open source ClickHouse



Agenda

Brief Introductions/Review:

- What is Distributed Tracing?
- What is OpenTelemetry?
- Why ClickHouse?

ClickHouse for Observability

Demo #1

Observability for ClickHouse

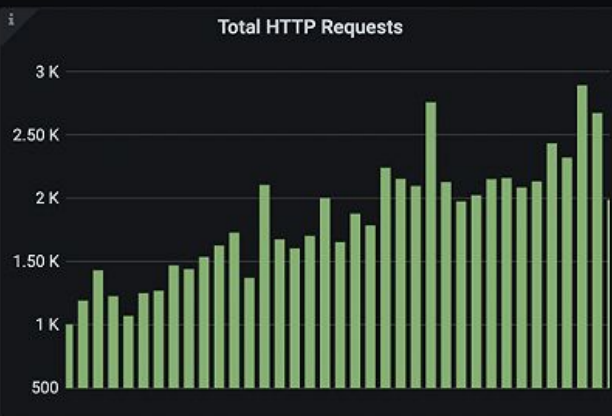
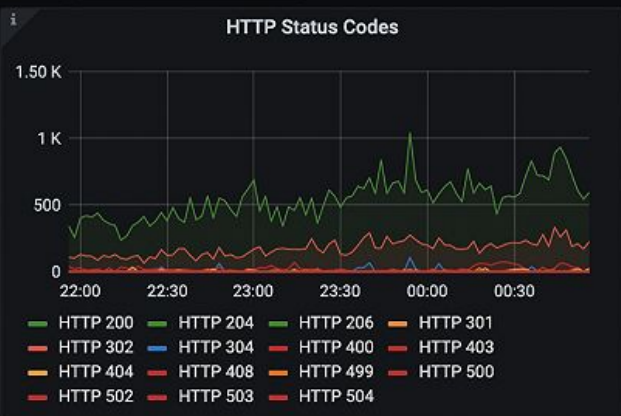
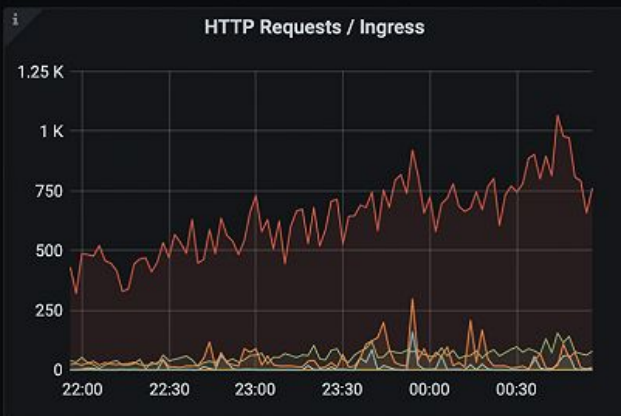
Demo #2!



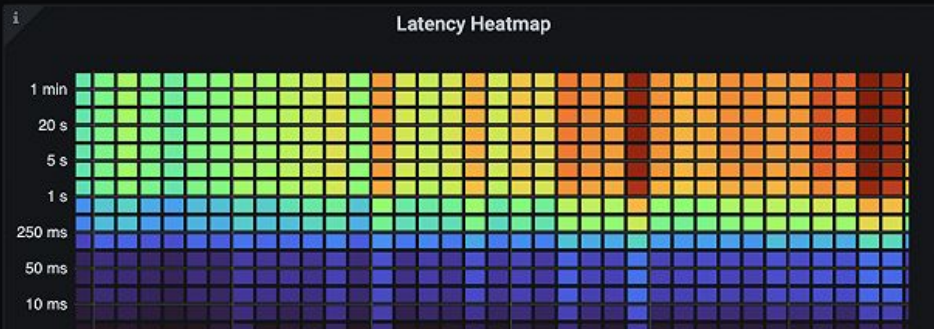
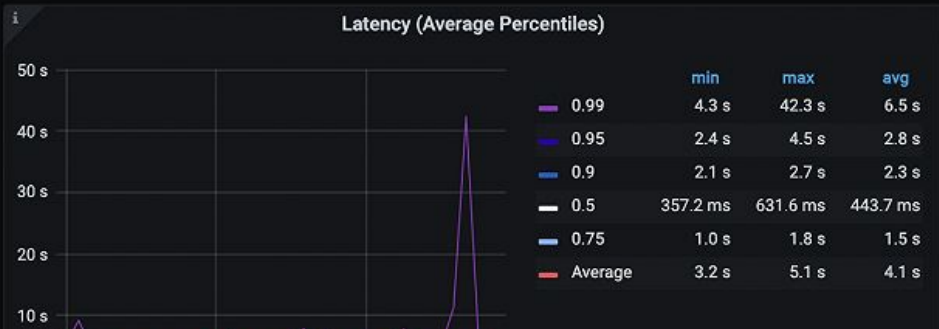
A humble log...

```
2024-07-01 09:35:34 231ms GET /home 200
```

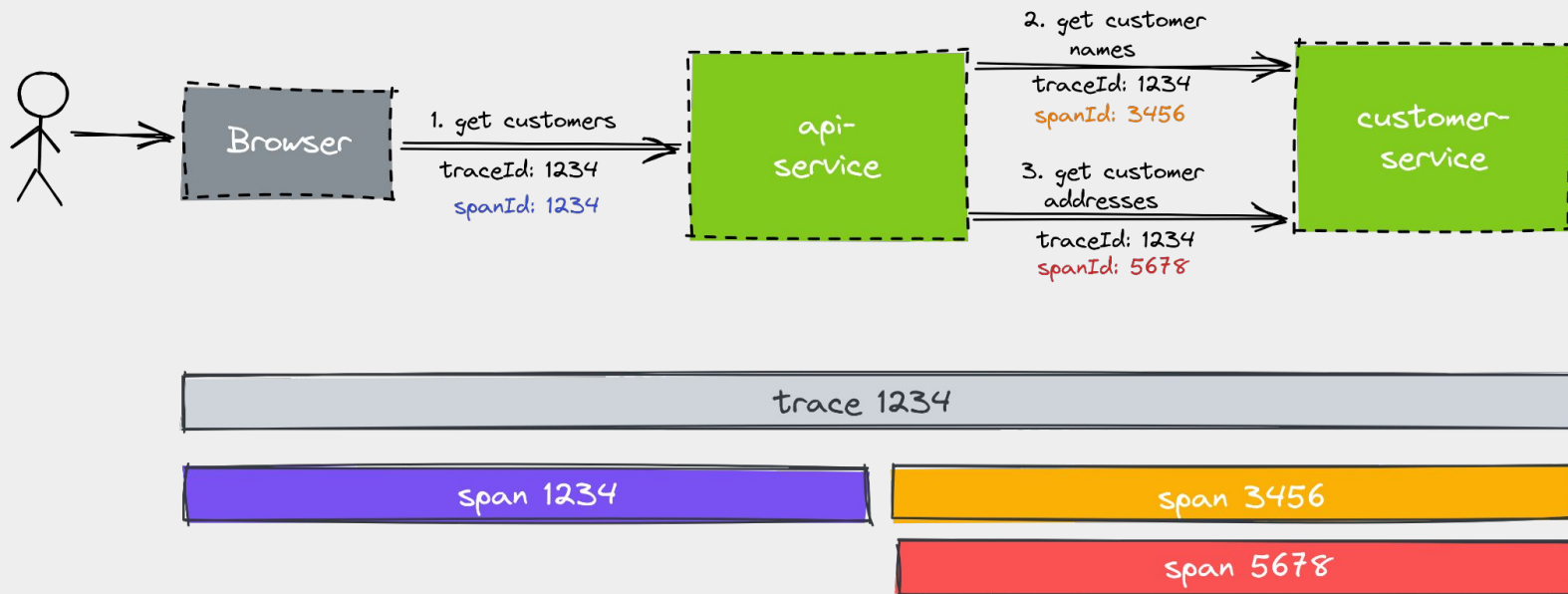

Overview



Latency



What is Distributed Tracing?



Our humble log...

```
2024-07-01 09:35:34 231ms GET /home 200
```


Make it a trace span:

```
2024-07-01 09:35:34 spanId:4f2c traceId:214a  
231ms GET /home 200
```

▼ frontend: HTTP GET ca28836

Find...

🔍 ⬆ ⬇ ✕

🔗

Trace Timeline

Start May 31 2023, 11:36:59.537 Duration 24.95ms Services 4 Depth 7 Total Spans 17



Event & Operation	0μs	6.24ms	12.48ms	18.71ms
frontend HTTP GET				
frontend grpc.oteldemo.RecommendationService/List...				17.69ms
▼ recommendationservice /oteldemo.Recomm...			14.12ms	
▼ recommendationservice get_product_list			13.61ms	
▼ recommendationservice /oteldemo...			11.54ms recommendationservice::oteldemo.FeatureFlagService/GetF...	
▼ featureflagservice /oteldemo.Fe...		2.23ms		
featureflagservice featurerefl...		1.93ms		
▼ recommendationservice /oteldemo...			1.62ms	
productcatalogservice otelde...			29μs	
frontend grpc.oteldemo.ProductCatalogService/GetPr...				5.48ms
productcatalogservice oteldemo.ProductCat...				27μs
frontend grpc.oteldemo.ProductCatalogService/GetPr...				4.42ms
productcatalogservice oteldemo.ProductCat...				9μs
frontend grpc.oteldemo.ProductCatalogService/GetPr...				5.44ms
productcatalogservice oteldemo.ProductCat...				21μs
frontend grpc.oteldemo.ProductCatalogService/GetPr...				3.16ms
productcatalogservice oteldemo.ProductCat...				20μs

Observability is not any one signal

Metrics

Aggregable

Is there a problem?

Traces

Request-Scoped

Where is the problem?

Logs

Verbose, time-stamped records

What is the problem?

Distributed Tracing is the *“Killer App”*

Understand
complete request
flows

Create a real-time
map of system
topology and
dependencies

Derive metrics
from the richness
of trace metadata

Enrich logs and
metrics with
context

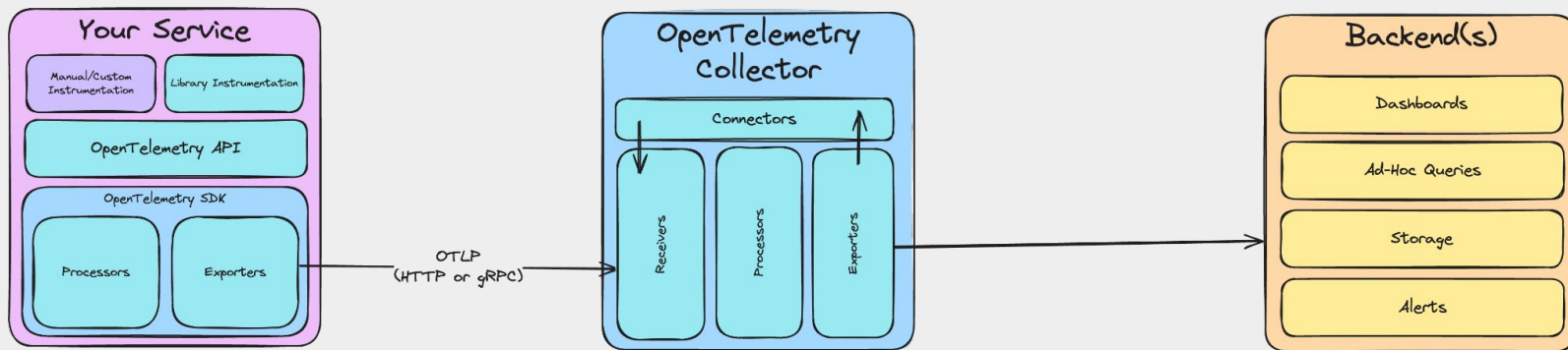
Introducing OpenTelemetry

The background image shows a large, white, dome-shaped observatory at night. The sky is dark blue with many stars visible. In the foreground, several people are gathered around the observatory, some looking through telescopes. There are some small, brightly lit structures and equipment near the observatory. The overall scene is illuminated by warm, orange light, possibly from the observatory's interior or nearby lights.

What is OTel?

- Specifications:
 - OTLP
 - Semantic Conventions
 - W3C Trace Context
- Libraries & Tools:
 - Language-specific SDKs & APIs
 - Instrumentation Libraries
 - The OpenTelemetry Collector
 - Extensions
- Community
 - Vendor-neutral, Open Source
 - 2nd most active CNCF project

What is OpenTelemetry?





What is ClickHouse?

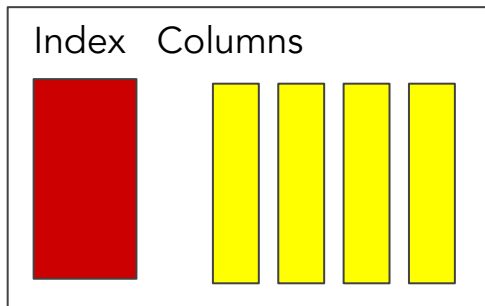
- Apache 2.0
- SQL Compatible
- Columnar
- Really really fast
- Petabyte scale (for now)
- Eats cardinality for breakfast



**Why
ClickHouse?**

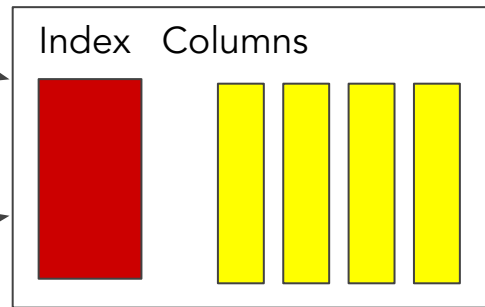
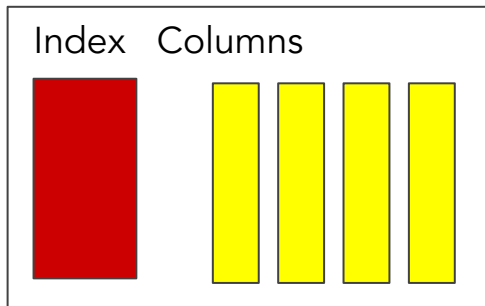
Telemetry is
WORM:
Write-Once
Read-Many

Part

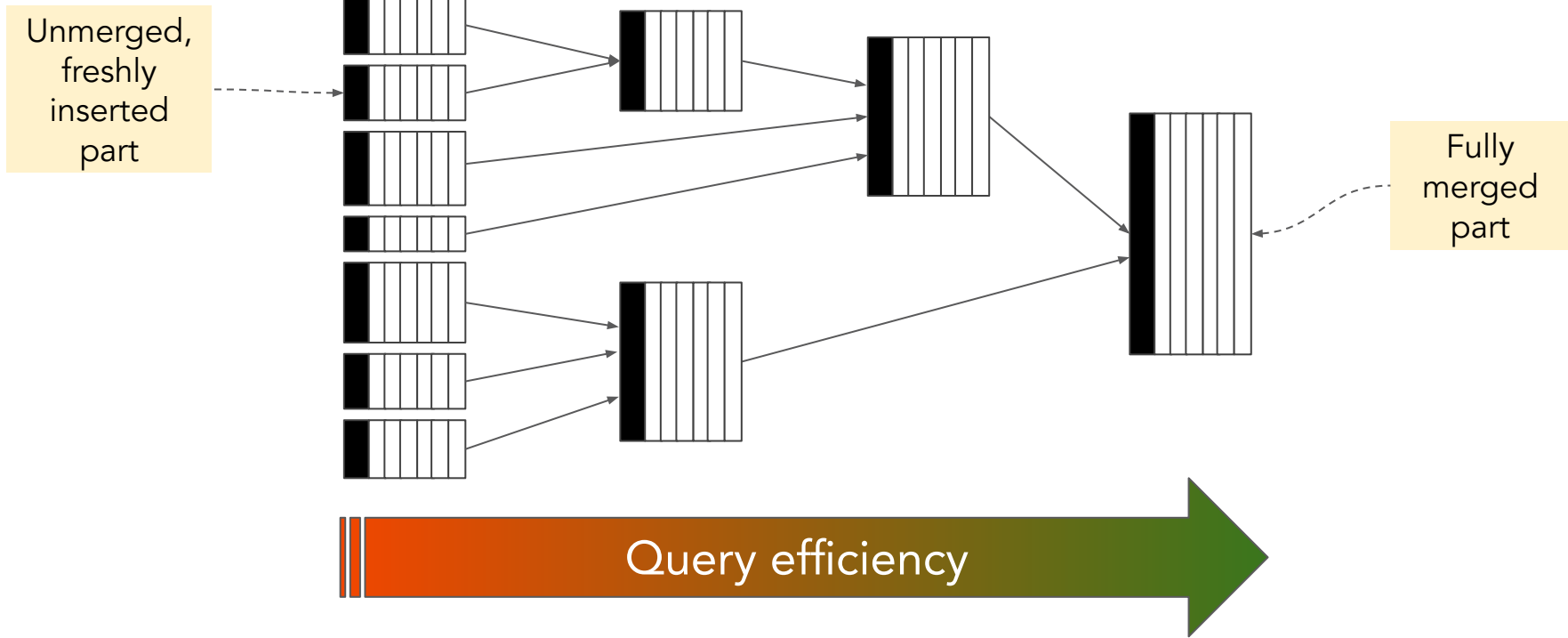


Rewritten, Bigger Part

Part



Update and delete also rewrite parts





More Benefits

- Time-friendly
- Easy cleanup
- TTL & tiered storage
- Extreme cardinality
- Excellent compression
- Flexible schema

Observability with ClickHouse

Storing and querying distributed traces from applications

Observability Integrations

OpenTelemetry Collector Exporter

Grafana Datasource

Kafka Connector

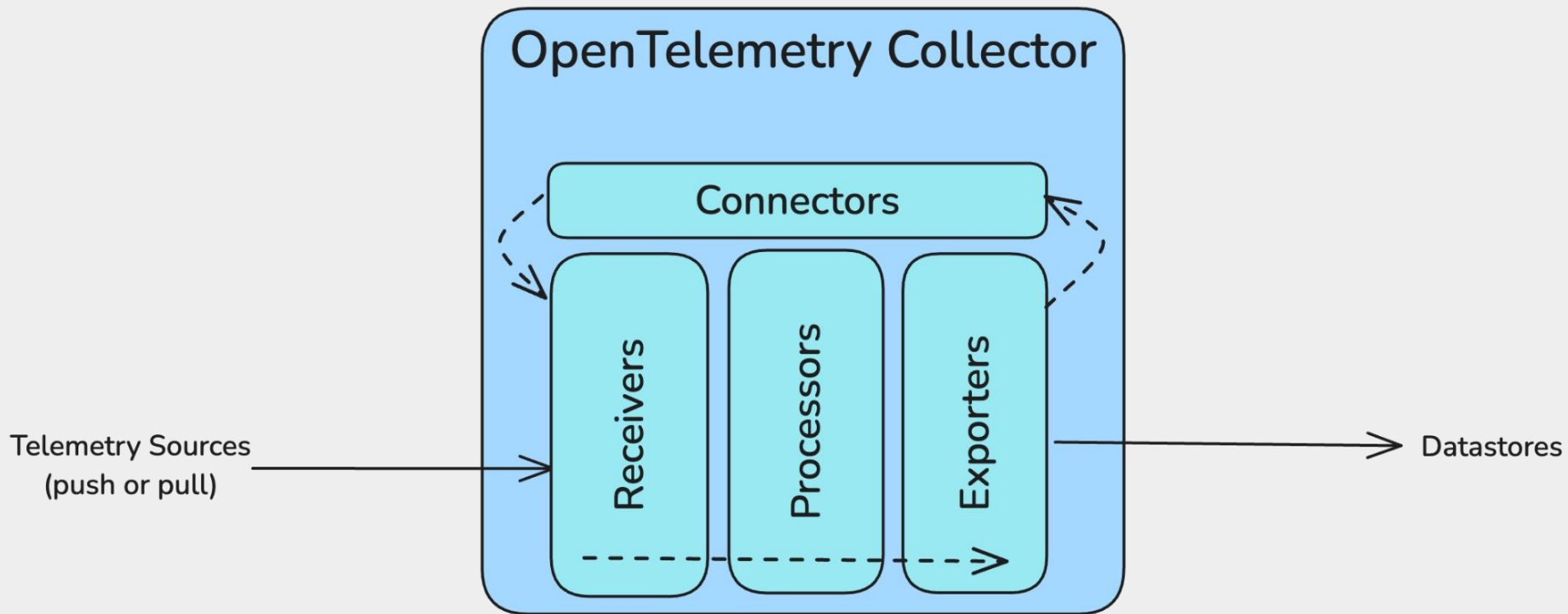
Coroot, SigNoz, ClickStack

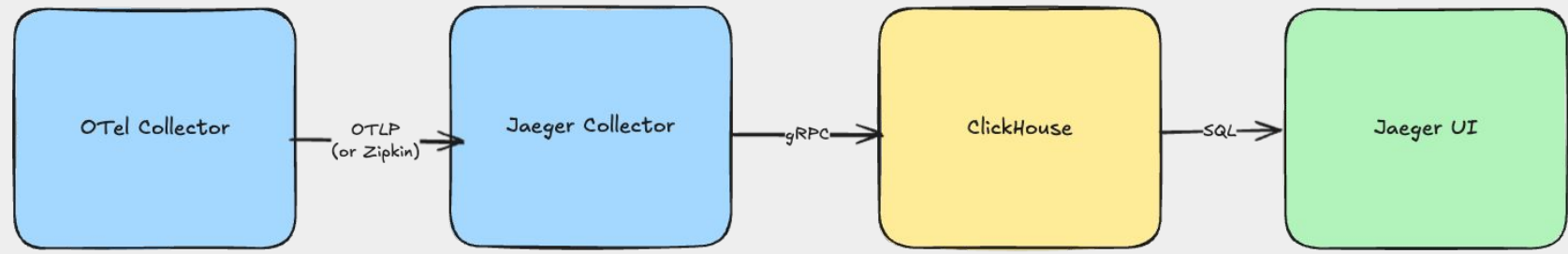
Quesma

Iceberg + Parquet...

The OpenTelemetry Collector

A night scene at an observatory. A large, white, dome-shaped telescope building is the central focus, with its entrance open and illuminated from within. Several people are standing near the entrance, some looking through telescopes. To the left, there are smaller, rectangular structures with open lids, possibly for equipment storage. The sky is dark blue with many stars visible. The foreground shows a paved path and some low-lying vegetation.





frontend: HTTP GET ca28836

Find...

Start May 31 2023, 11:36:59.537 Duration 24.95ms Services 4 Depth 7 Total Spans 17



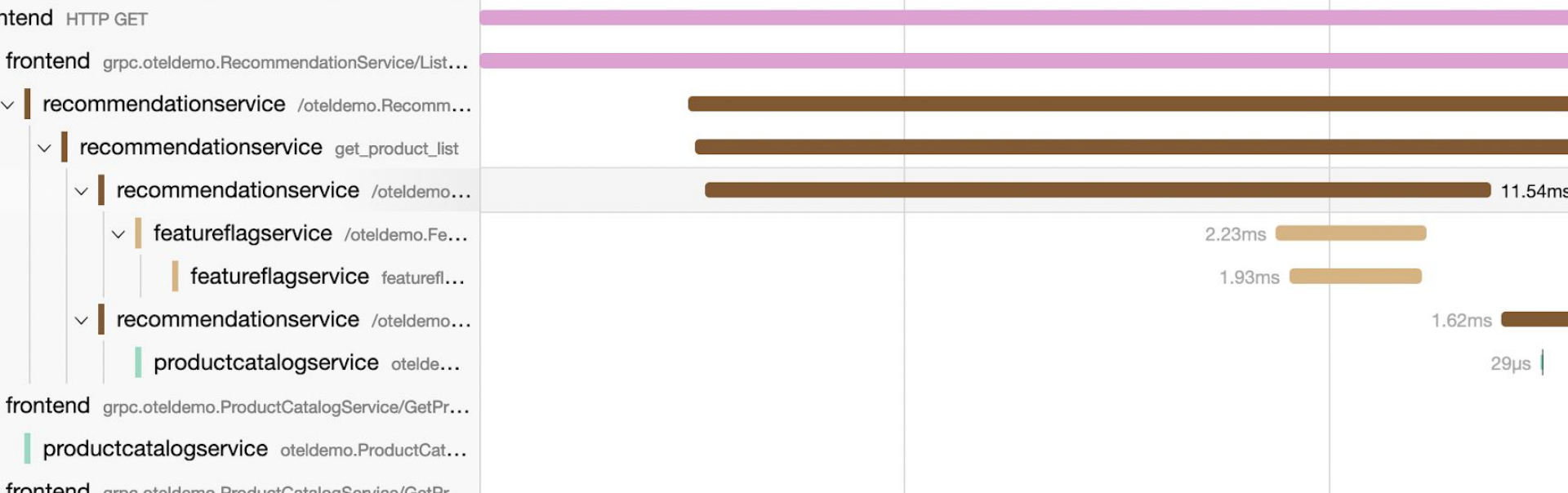
Event & Operation

v > v v >> ||

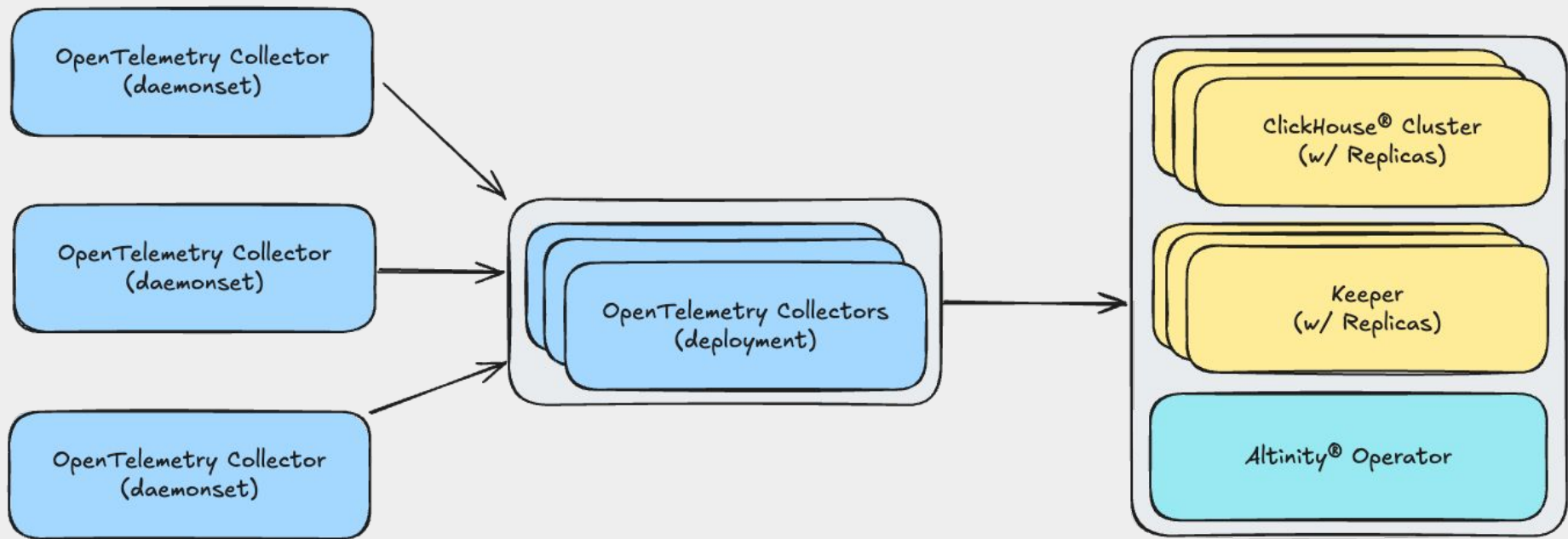
0µs

6.24ms

12.48ms

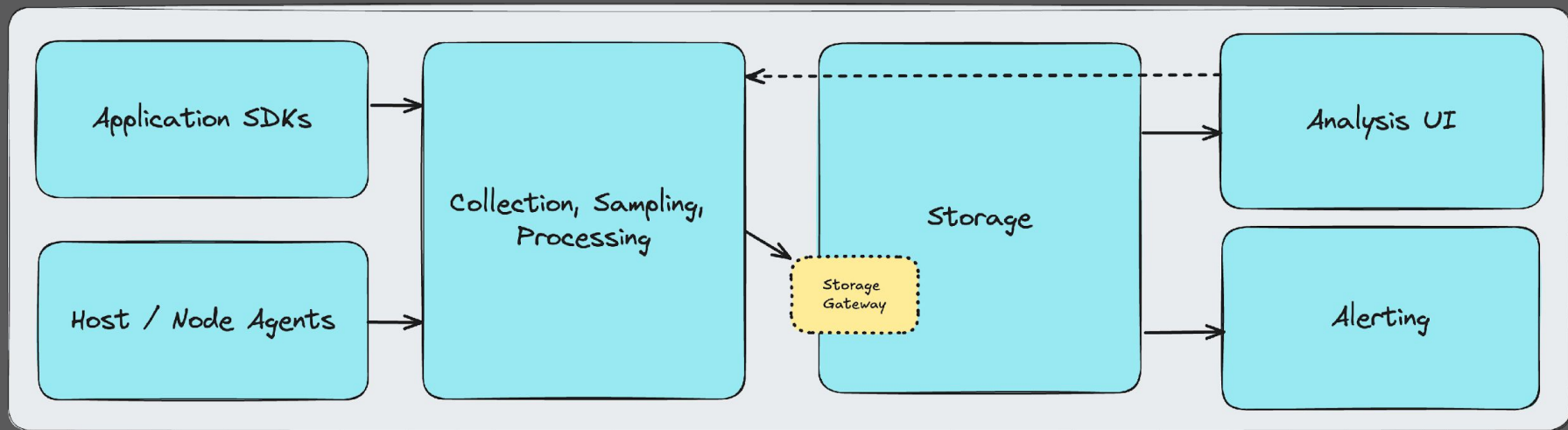






Demo

Traces with ClickHouse + OpenTelemetry



Overview

HEALTH

SERVICE MAP

TRACES

NODES

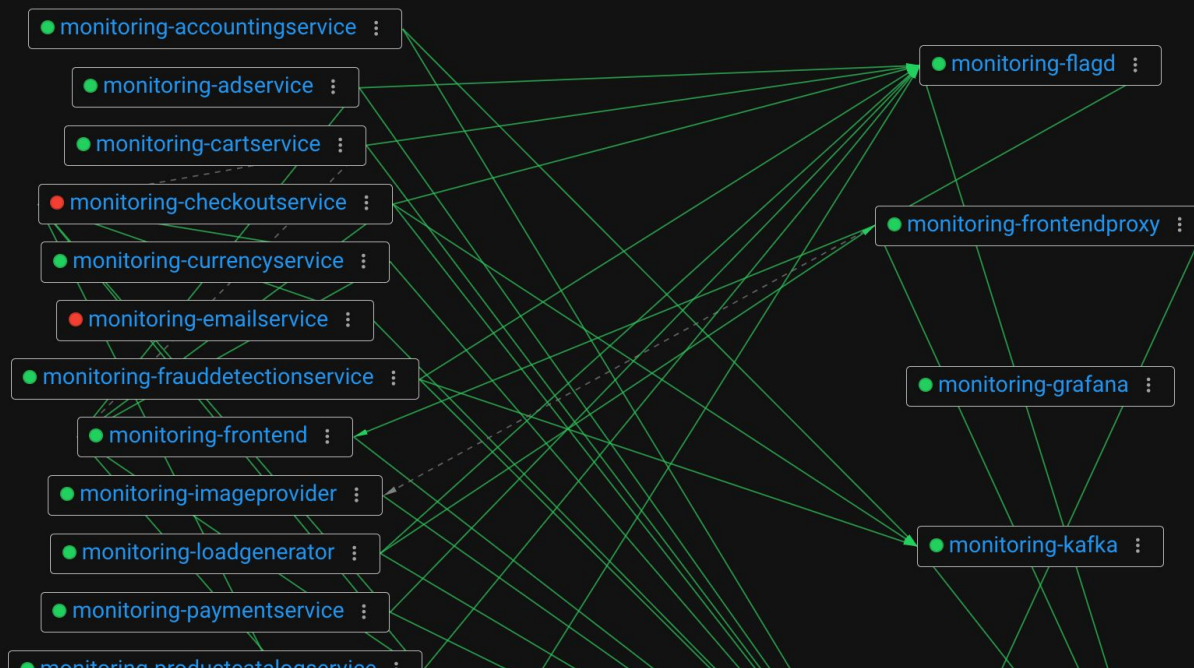
DEPLOYMENTS

COSTS

🔍 search

namespaces

monitoring ✕ ▾

☒ application ☐ control-plane ☒ monitoring +



ClickStack®

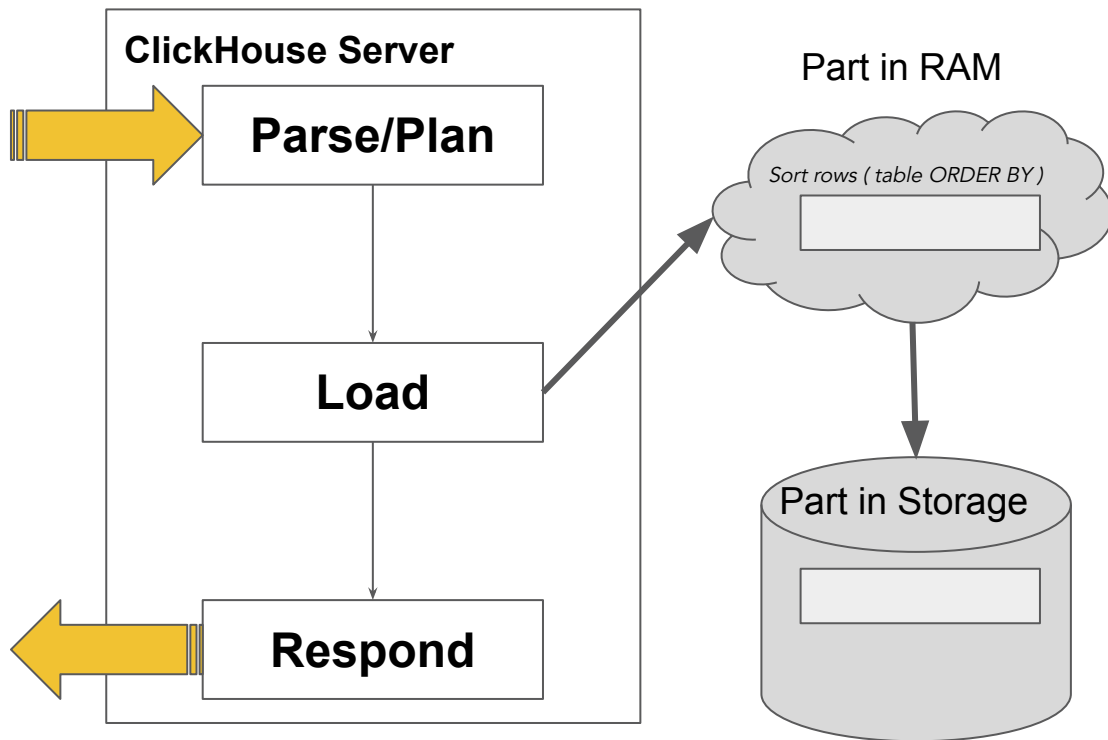
ClickHouse Architecture

How queries are processed

ClickHouse Architecture: Processing an Insert

```
INSERT INTO sdata  
VALUES  
(15, 'TEMP', . . .),  
(15, 'TEMP', . . .)
```

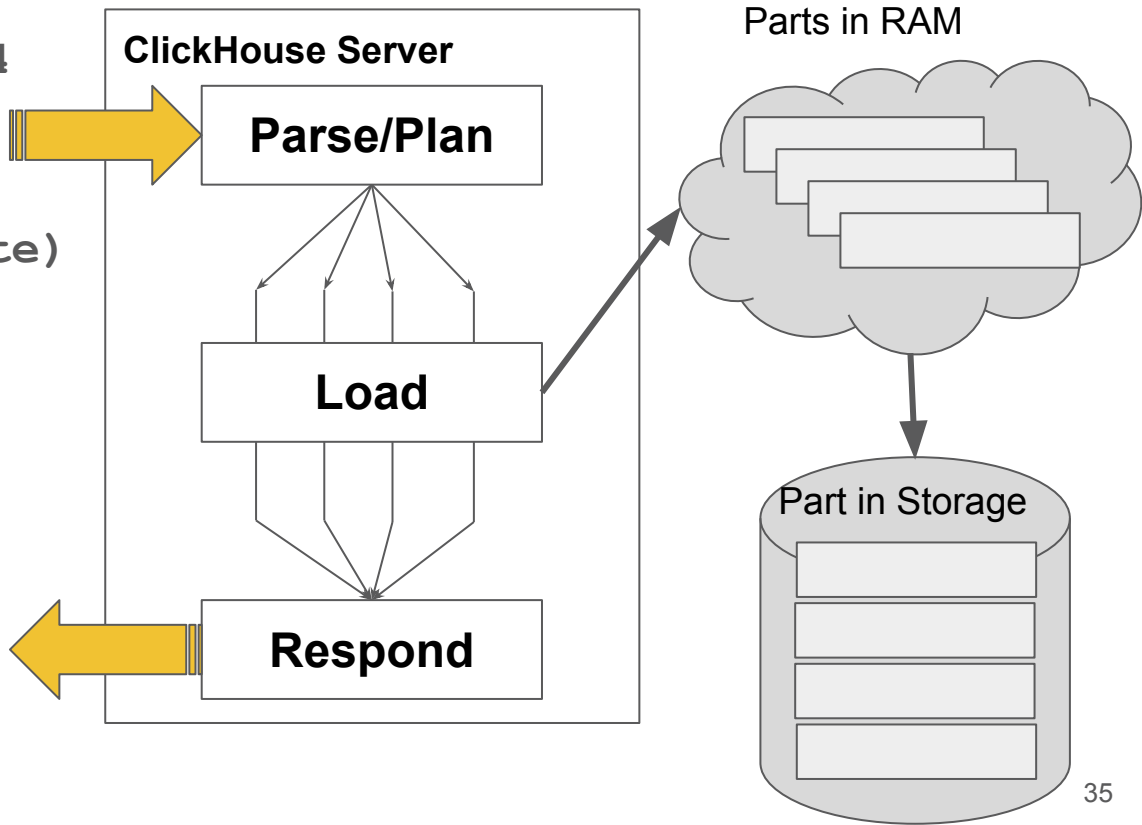
```
2 rows in set. Elapsed: 0.003 sec.
```



How can we make this more efficient? Parallelize!

```
set max_insert_threads=4  
insert into ontime_test  
select * from ontime  
  where toYear(FlightDate)  
        between 2000 and 2001
```

```
2 rows in set. Elapsed: 0.003 sec.
```

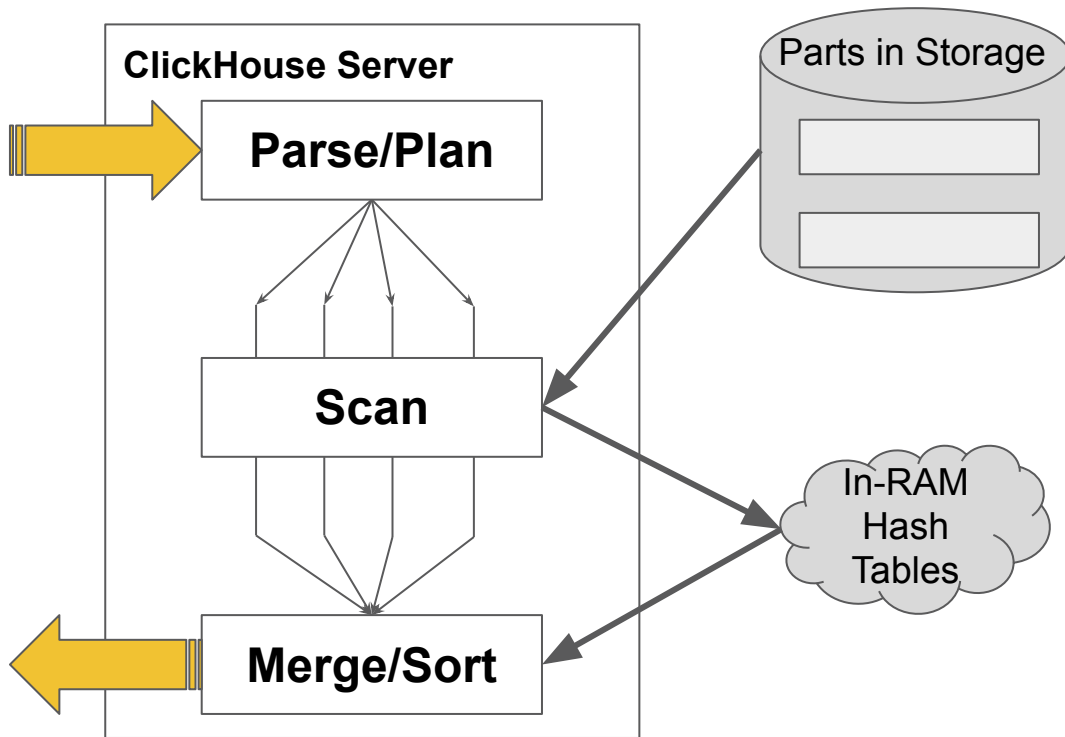


How does ClickHouse process a query with aggregates?

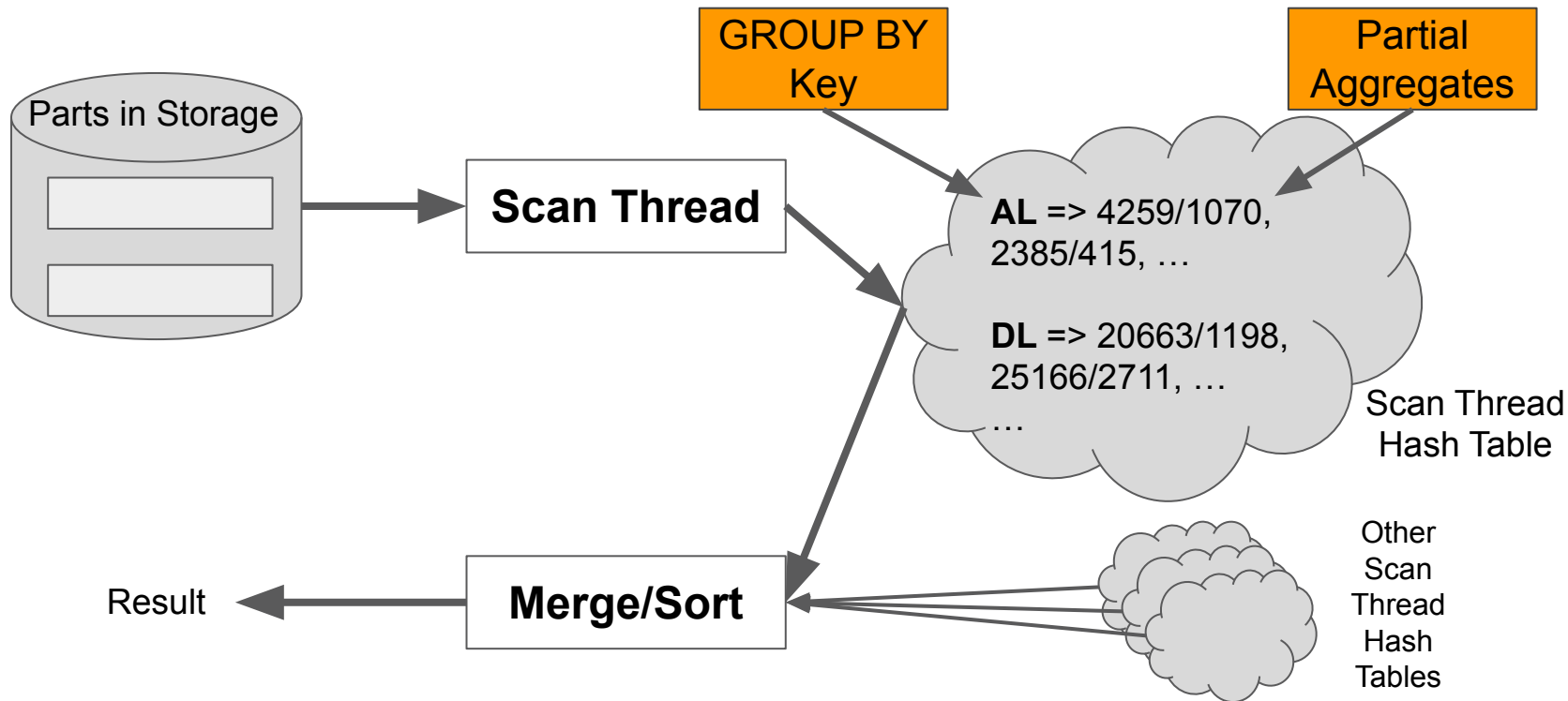
```
SELECT Carrier,  
       avg(DepDelay)AS Delay  
FROM ontime  
GROUP BY Carrier  
ORDER BY Delay DESC
```

Carrier	Delay
B6	12.058290698785067
EV	12.035012037703922
NK	10.437692933474269

...



How does a ClickHouse thread do aggregation?



Observability for ClickHouse

Built-in metrics, logs, and *traces*



System Tables

- Metric Log
- Part Log
- Query Log
- Query Views Log
- Trace Log
- Distributed Traces*

```

SHOW CREATE TABLE system.opentelemetry_span_log;
---
CREATE TABLE system.opentelemetry_span_log
(
  `hostname` LowCardinality(
String) COMMENT 'The hostname where this span was captured.',
  `trace_id` UUID COMMENT
  'ID of the trace for executed query.',
  `span_id` UInt64 COMMENT
  'ID of the trace span.',
  `parent_span_id` UInt64 COMMENT
  'ID of the parent trace span.',
  `operation_name` LowCardinality(
String) COMMENT 'The name of the operation.',
  `kind` Enum8(
  'INTERNAL' = 0, 'SERVER' = 1, 'CLIENT' = 2, 'PRODUCER' = 3, 'CONSUMER' = 4) COMMENT 'The SpanKind of the span. INTERNAL - Indicates that the
span represents an internal operation within an application. SERVER - Indicates that the span covers server-side handling of a synchronous RPC
or other remote request. CLIENT - Indicates that the span describes a request to some remote service. PRODUCER - Indicates that the span
describes the initiators of an asynchronous request. This parent span will often end before the corresponding child CONSUMER span, possibly
even before the child span starts. CONSUMER - Indicates that the span describes a child of an asynchronous PRODUCER request.',
  `start_time_us` UInt64 COMMENT
  'The start time of the trace span (in microseconds).',
  `finish_time_us` UInt64 COMMENT
  'The finish time of the trace span (in microseconds).',
  `finish_date`
Date COMMENT 'The finish date of the trace span.',
  `attribute` Map(LowCardinality(
String), String) COMMENT 'Attribute depending on the trace span. They are filled in according to the recommendations in the OpenTelemetry
standard.',
  `attribute.names` Array(LowCardinality(
String)) ALIAS mapKeys(attribute),
  `attribute.values` Array(
String) ALIAS mapValues(attribute)
)
ENGINE = MergeTree
PARTITION
BY toYYYYMM(finish_date)
ORDER BY (finish_date, finish_time_us, trace_id)
SETTINGS index_granularity =
8192
COMMENT
'Contains information about trace spans for executed queries.'

1 row in set. Elapsed: 0.034 sec.

```

Built-in Spans Table

System table created dynamically

Enable tracing via setting (on query or user profile) or request headers

Using a TraceID Header:

```
$ clickhouse-client \  
  --opentelemetry-traceparent \  
  "00-4bf92f...929d0e0e4736-00f067aa0ba902b7-01"
```

Always Initialize a Trace:

```
SET opentelemetry_start_trace_probability = 1;
```

Demo

Tracing ClickHouse Queries

2025-06-05 16:45:46.500

Span Filters ⓘ

Service Name	=	All service names		
Span Name	=	All span names		
Duration	>	e.g. 100ms, 1.2s	<	e.g. 100ms, 1.2s
Tags	ⓘ	Select tag	=	Select value

Clear



Show matches only

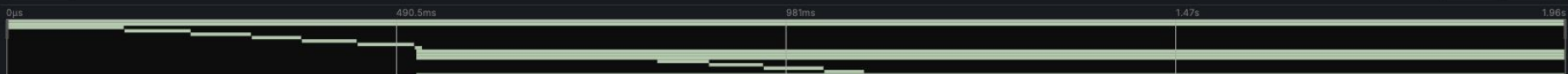


Show critical path only

17 spans ⓘ

Prev

Next



Service & Operation



0us490.5ms981ms1.47s1.96s

ClickHouse TCPHandler (1.96s)

query (1.96s)

- ListObjectS3 (146ms)
- ListObjectS3 (83ms)
- ListObjectS3 (76ms)
- ListObjectS3 (62ms)
- ListObjectS3 (69ms)
- ListObjectS3 (71ms)
- DB::InterpreterSelectQueryAnalyzer::execute() (1ms)

QueryPullPipeEx (1.44s)

PipelineExecutor::execute() (1.44s)

QueryPipelineEx (1.44s)

- ListObjectS3 (65ms)
- ListObjectS3 (68ms)
- ListObjectS3 (75ms)
- ListObjectS3 (50ms)

QueryPipelineEx (1.44s)





Client-Side Tracing

Supported ClickHouse Clients:

- Java
- Golang
- Python

Call to action!

Please help improve the OpenTelemetry
Auto-Instrumentation Libraries



Final Thoughts

- Distributed tracing is awesome (for applications especially)
- Consider ClickHouse as a massively scalable telemetry datastore
- Consider CH-based O11y tools for their performance with high cardinality signals
- Built-in observability is awesome



Blog post comparing
Observability Platforms

Thank you! Questions?

Contact us to learn more and join our
community:

<https://altinity.com>

<https://altinity.com/slack>

<https://github.com/Altinity/tracing-clickhouse>