



A Roundup of Observability Datastores

Josh Lee • Open Source Advocate @ Altinity • FOSDEM 2026

**"A Developer will never ask you,
'Hey, what filesystem is that?'"**

— Patrick McFadin





Josh Lee

Open Source Advocate
Altinity

*ClickHouse® is a registered trademark of ClickHouse, Inc.
Altinity is not affiliated with or associated with ClickHouse, Inc.
We are but humble open source contributors*



**Observability = Visibility +
Understanding**



50x

Observability data vs system data



What are we storing?

Metrics, Traces, Logs, Profiles, Events

Labels/Tags

Resource Metadata

Graphs & Topologies

Snapshots & Deltas

Configuration (e.g. alerts, users, dashboards)



What do we need for observability?

Fast streaming writes

Efficient compression & storage

Time-oriented management

“Real-time” analytics



"Anything you can do with a group by, that's what analytics is"

—Peter Marshall



More Requirements

Fast multi-row analytics

Full-text search

Tag/label search

Fast, frequent "last point" reads

Updates?



Database Archetypes

OLTP

OLAP

TSDB

Search/Analytics



Introducing the cast of characters

Postgres (OLTP)

Cassandra (OLTP)

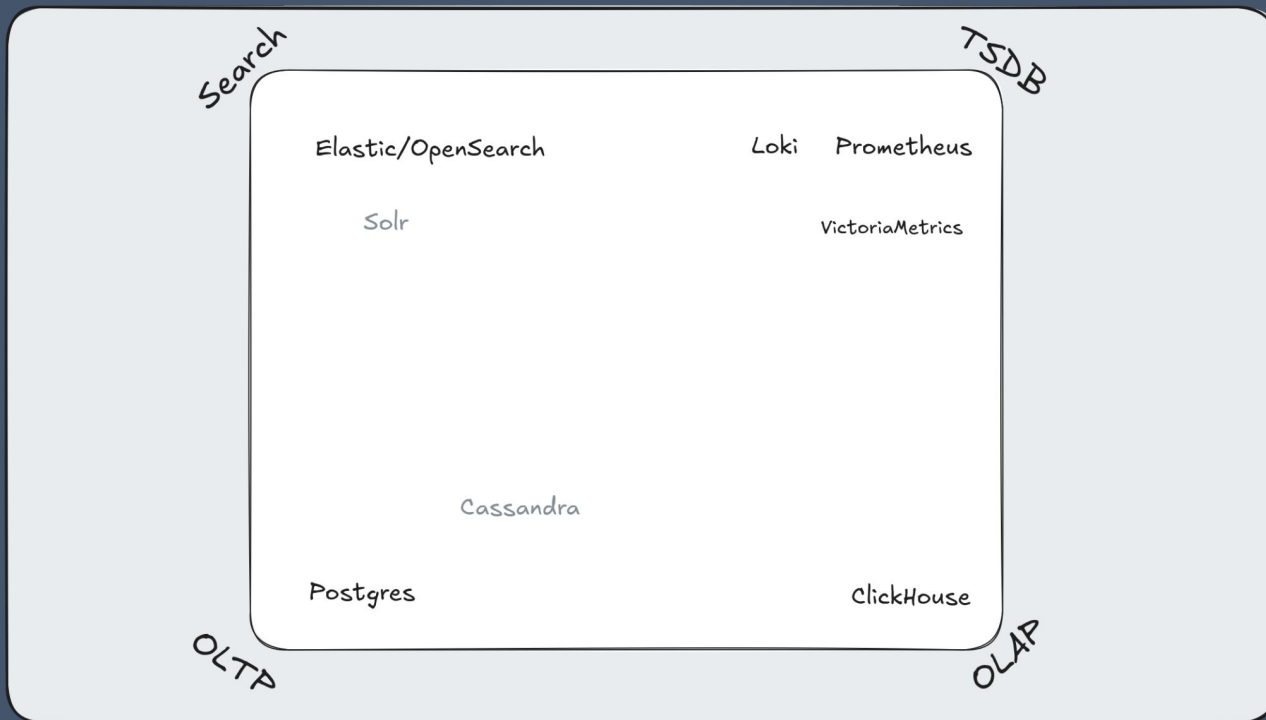
Elastic/OpenSearch (Search & Analytics)

Prometheus (TSDB)

ClickHouse (OLAP)



Taxonomies are challenging



Storage on disk



Database Storage Styles

Heap Pages + Commit Log

Time-series Blocks

Parts / Segments



How to make a Postgres

WAL

Heap Pages + MVCC (Multi-view Concurrency Control)

B-Tree Indexes



Postgres/MySQL/etc.

Optimized for updates/upserts and row-level reads

Strong ACID guarantees

Scaling horizontally is challenging

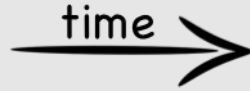


Analytics & Search Architecture

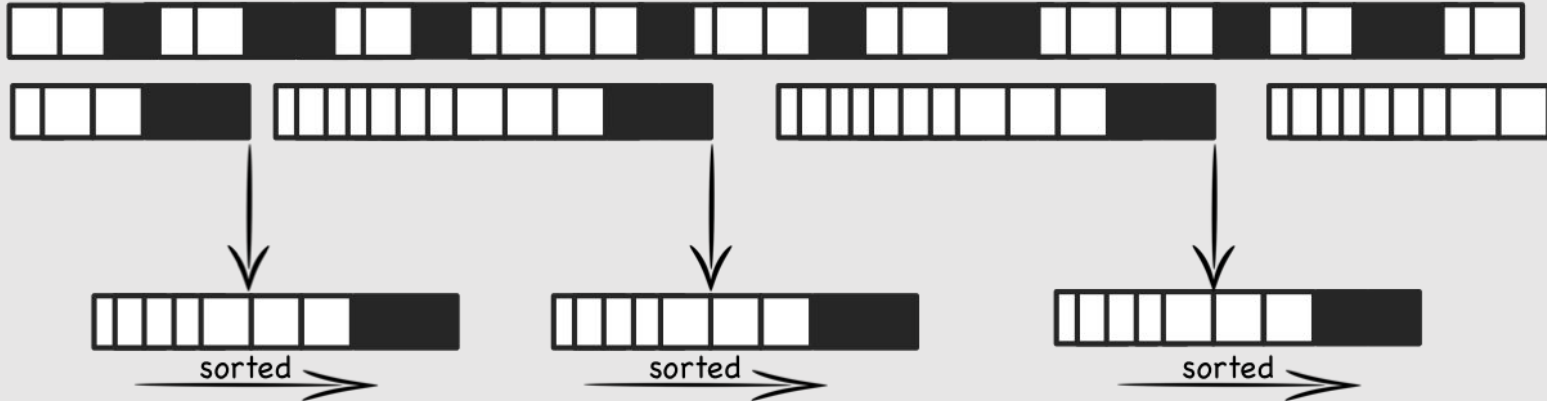


Log-Structured Merge Tree

<http://www.benstopford.com/2015/02/14/log-structured-merge-trees/>



Data stream of k-v pairs ...are buffered in sorted memtables



Lucene Family:

**Cassandra, Elastic/OpenSearch,
Apache Solr, Splunk, and
more...**

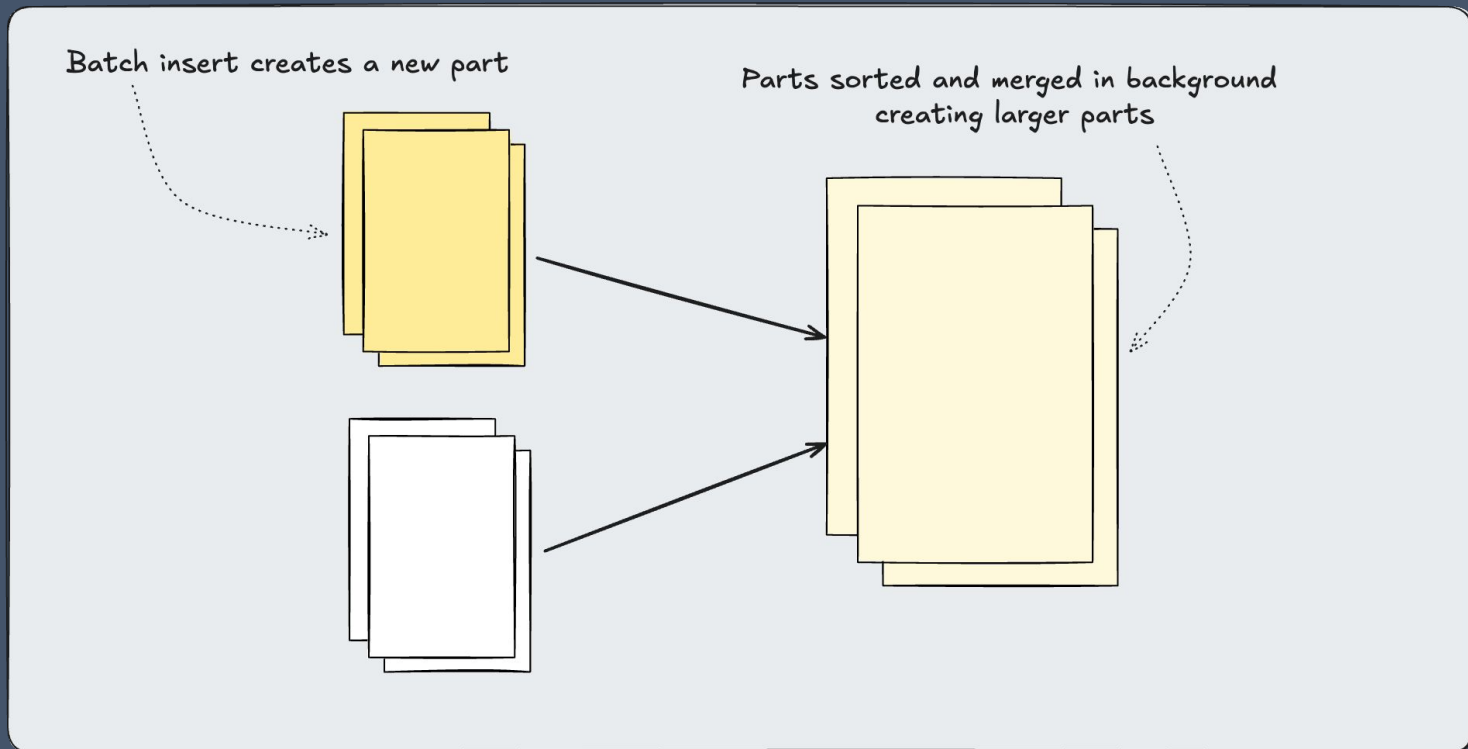


Immutable Parts / Segments

w/ Background Compaction



Immutable Parts / Segments



Cassandra

Wide-event Scalable OLTP



Vector Engines & Search

Inverted Indexes

Bloom Filters

Approximate Nearest Neighbor (ANN Graph)



Inverted Indexes

```
"cat" → [doc1, doc3, doc7]  
"dog" → [doc2, doc5]  
"parrot" → [doc1, doc4]
```



Bloom Filters - “so call me maybe”

No false negatives, but will result in mild overscans

```
"otel-collector-prod-01" → 31  
"otel-collector-prod-10" → 31
```



Bloom Filters - “so call me maybe”

No false negatives, but will result in mild overscans

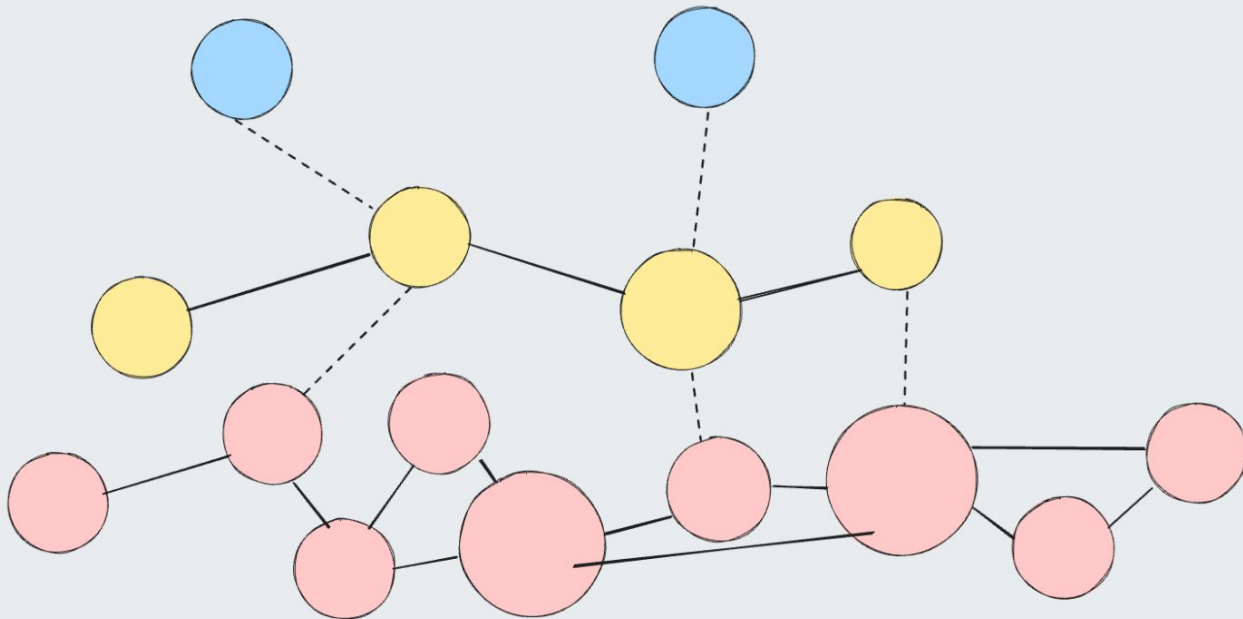
```
"otel-collector-prod-01" → 31  
"otel-collector-prod-10" → 31
```

Much sparser than an inverted index - may fit entirely in memory



Approximate Nearest Neighbor (ANN)

A way to organize and filter vectors



Prometheus (& friends)

Time-series Database



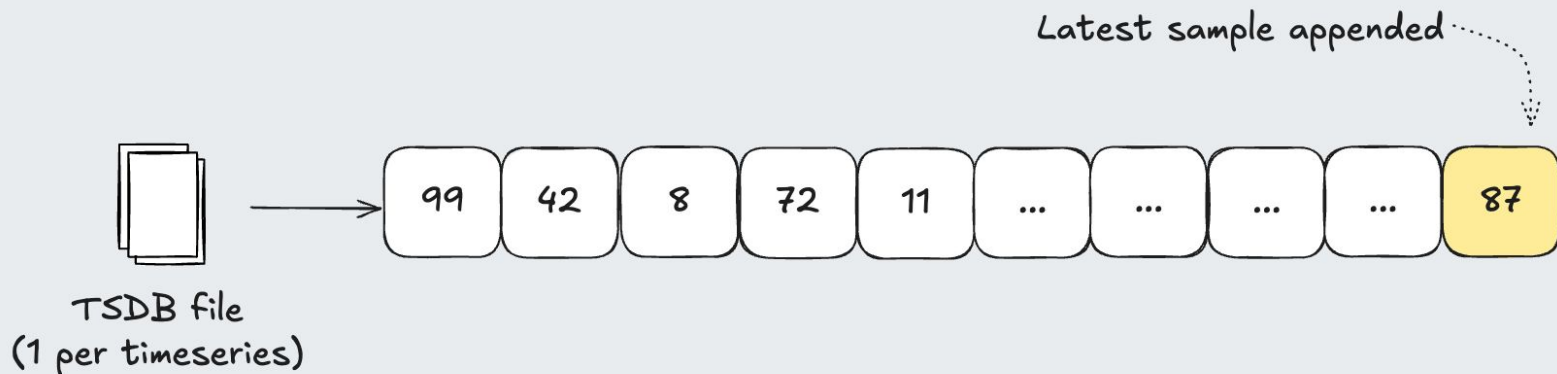
TSDB Blocks

Append-only



TSDB: Data is naturally ordered by time

Excellent for frequent reads of last-sample



TSDB

No. of time series = $\text{cardinality}^{\text{dimensionality}}$

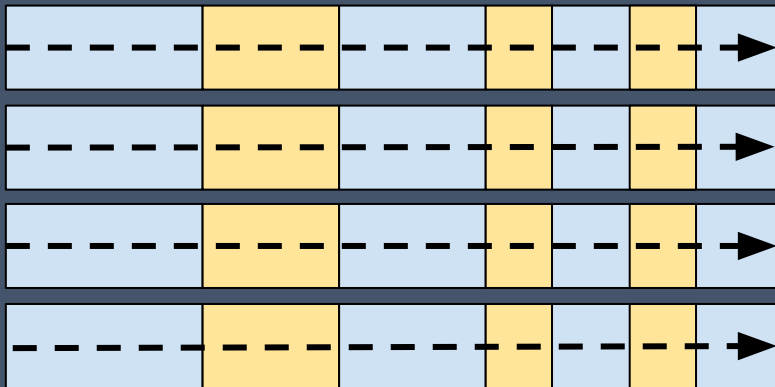


Row-oriented vs column-oriented storage



Row-oriented Storage

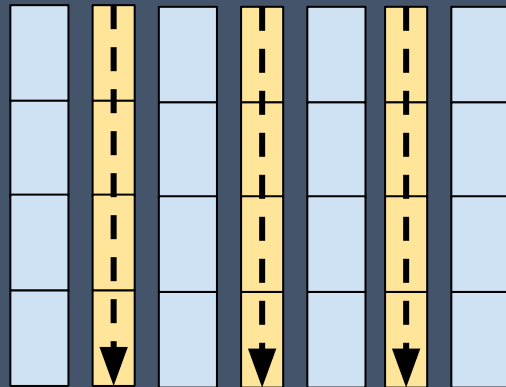
Read all columns in row



Rows compressed minimally or not at all

Column-oriented

Read only selected columns



Columns highly compressed

ClickHouse

Column-oriented MergeTree



59 GB
(100%)

Read 109
columns

Read 3 columns
from 109

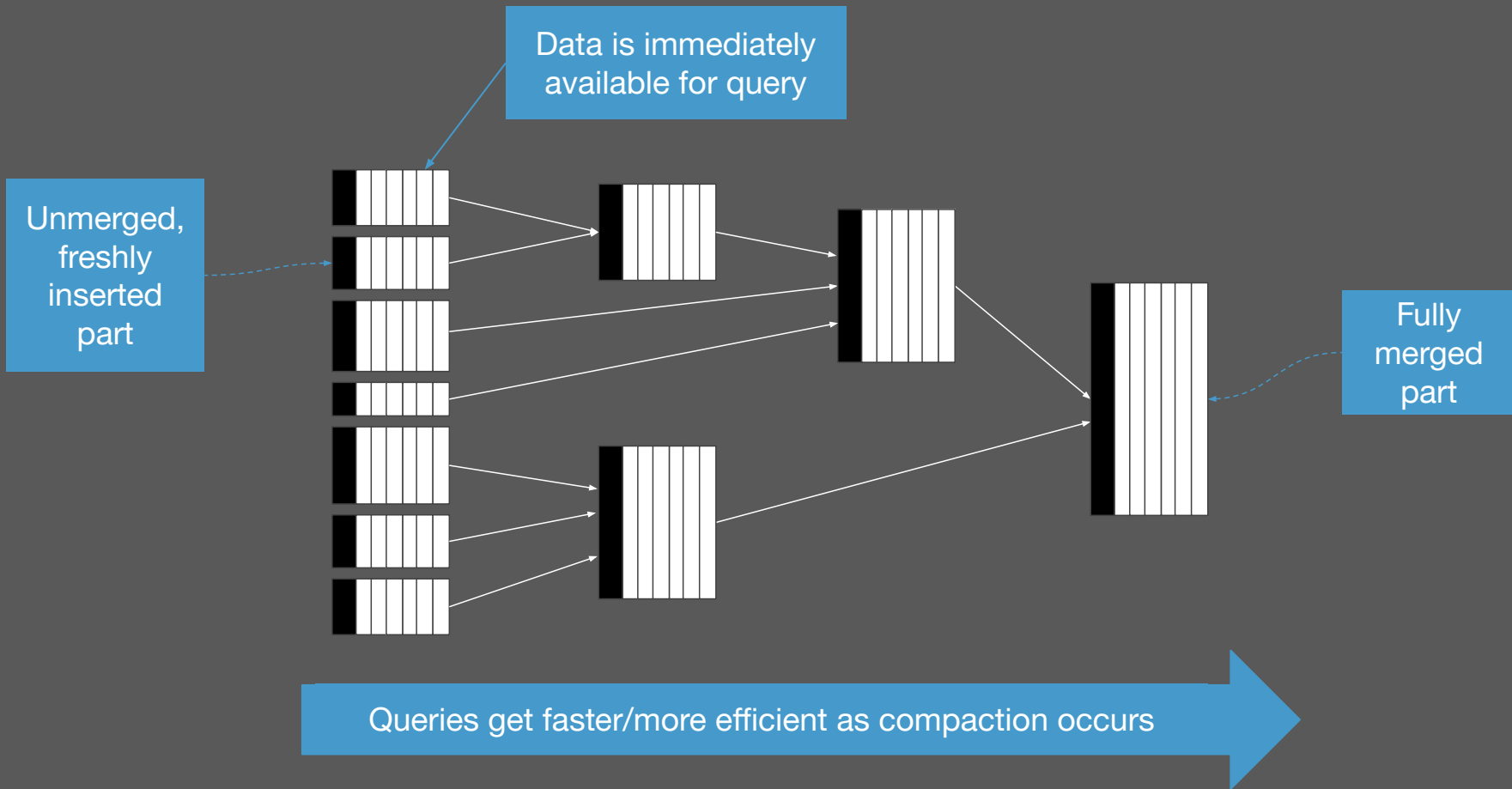
1.7 GB
(3%)

Read 3
compressed
columns

21 MB (.035%)

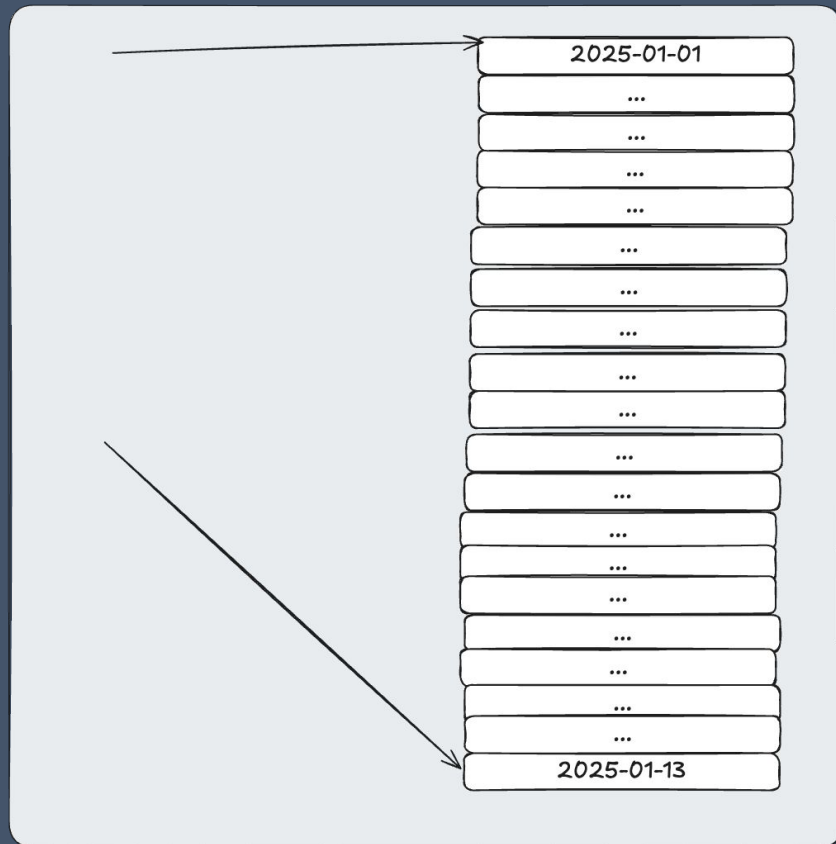
Read 3
compressed
columns over 8
threads

2.6 MB
(.0044%)



Sparse Indexes

Quickly & cheaply find data
based on an ordered key



```
graph LR; OT[OpenTelemetry Collector] --> CH[ClickHouse]; OT --> P[Prometheus]; OT --> OS[OpenSearch];
```

The diagram illustrates the OpenTelemetry Collector architecture. On the left, five green rounded rectangles represent data sources: OpenTelemetry Traces, SNMP Metrics, Prometheus Metrics, Container Logs, and Kubelet Logs. These sources feed into a central blue rounded rectangle labeled 'OpenTelemetry Collector'. Inside this collector, there are three yellow rounded rectangles representing pipelines: Traces Pipeline, Metrics Pipeline, and Logs Pipeline. A dashed arrow indicates a flow from the Traces Pipeline to the Metrics Pipeline. On the right, three cyan rounded rectangles represent data sinks: ClickHouse, Prometheus, and OpenSearch. Arrows show the data flow from the collector's pipelines to these sinks: Traces Pipeline to ClickHouse, Metrics Pipeline to Prometheus, and Logs Pipeline to OpenSearch.



VictoriaMetrics

TSDB meets MergeTree



Loki

Uncomplicated logging (with label indexes)



Honorable Mentions

Cortex, Thanos, Mimir, TimecaleDB, Solr, Druid...



At (very) small scale

Just use what you have until it breaks (Postgres)



Hooked-on full-text search

Elastic/OpenSearch has your back



One database for everything

ClickHouse is pretty cool



Wide-event analytics

ClickHouse is awesome



Lots of "last-sample" reads + alerts

Choose a TSDB like Prometheus or VictoriaMetrics



Database (Orientation)	Style/QL	Storage	Indexes	Use Case
Postgres (Row)	OLTP/SQL	Heap Pages	B-Tree	Update/upsert with guarantees
Cassandra (Document)	OLTP/CQL	Lucene Segments	Inverted	Scalable upserts
Prometheus (Columnar*)	TSDB/PromQL	TSDB files	By label	Time-series metrics, alerting
OpenSearch (Document)	Search/LuceneQL	Lucene Segments	Inverted, Bloom Filter, ANN	Full-text search
ClickHouse (Columnar)	OLAP/SQL	MergeTree Parts	Sparse, Inverted, and more...	Real-time analytics



Open Source Analytics Community Meetup

Monday, Feb 2nd @ Commons Hub

<https://osacom.io/events/>



**Thank you and happy
querying!**

