



A Roundup of Observability Datastores

Josh Lee • Open Source Advocate @ Altinity • Code Europe

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

— Patrick McFadin



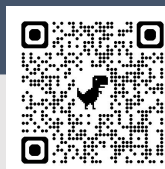


Josh Lee

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*

Josh Lee • <https://joshuamlee.com/events/code-europe-26>

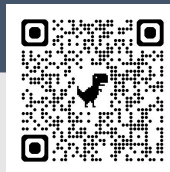


**Observability = Visibility +
Understanding**



50x

Observability data vs system transaction data



What are we storing?

Metrics, Traces, Logs, Profiles, Events

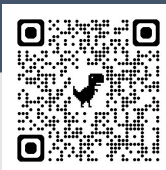
Labels/Tags

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)

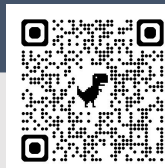
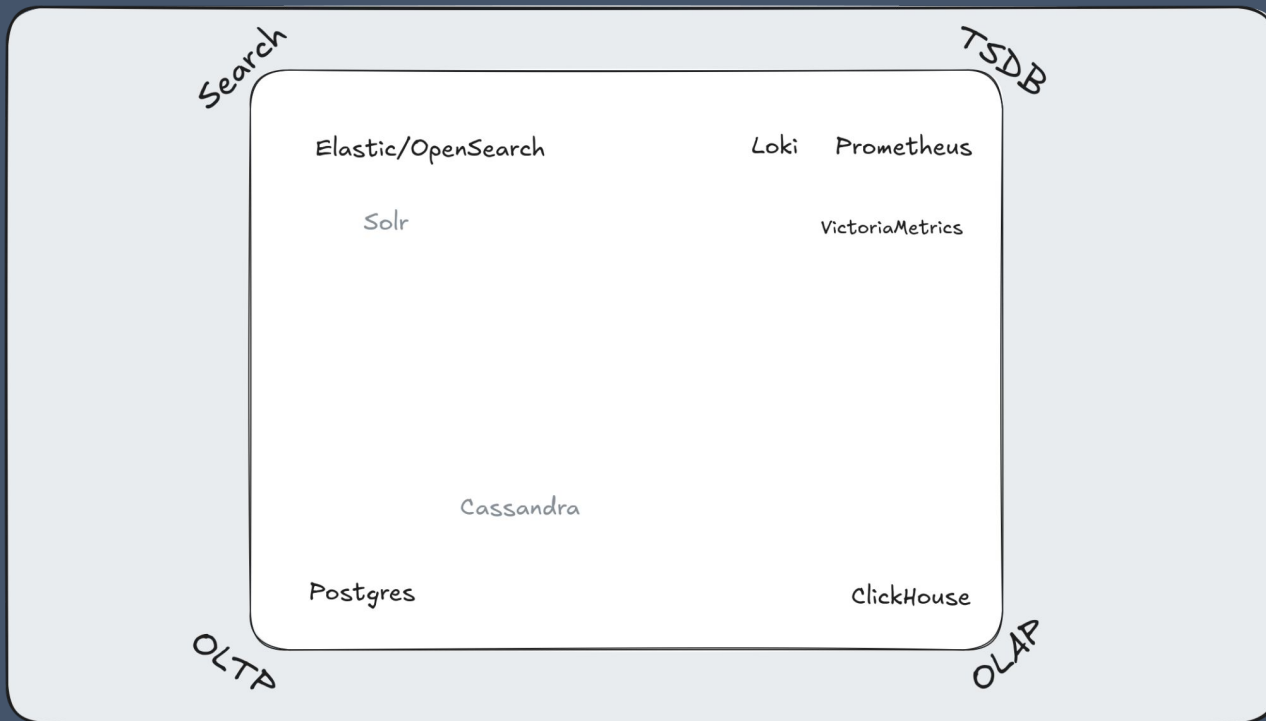
Elastic/OpenSearch (Search & Analytics)

Prometheus (TSDB)

ClickHouse (OLAP)



Taxonomies are challenging



Storage on disk



Database Storage Styles

Heap Pages + Commit Log

Time-series Blocks

Parts / Segments



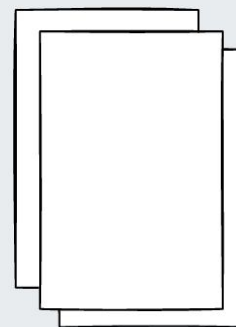
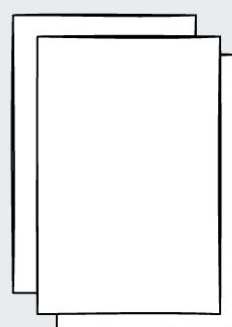
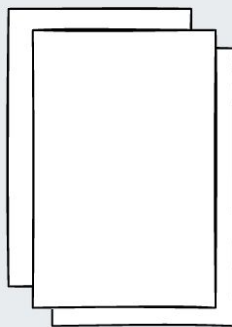
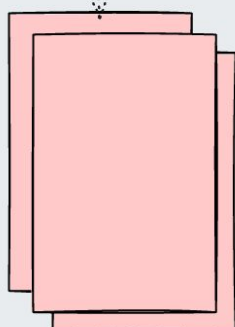
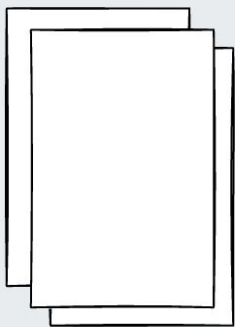
Heap Pages

* the JBOD of storage styles



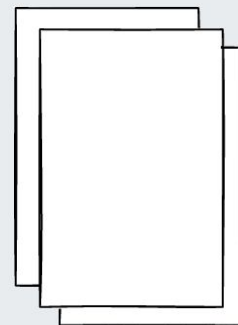
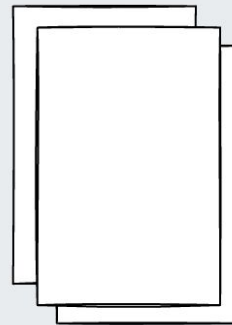
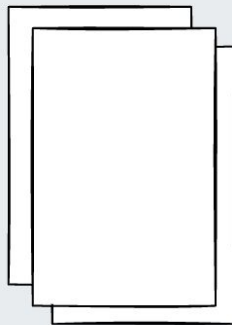
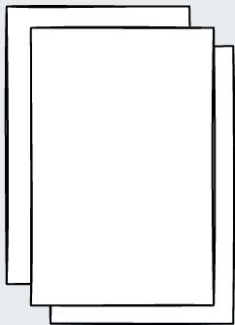
Heap Pages

Page marked for deletion



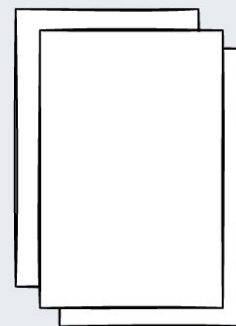
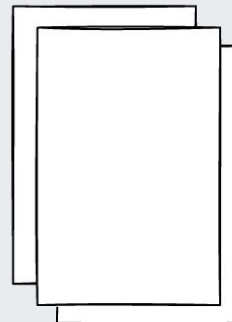
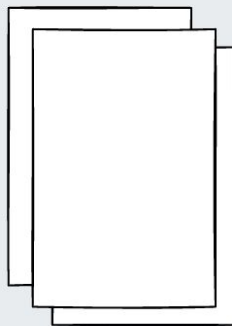
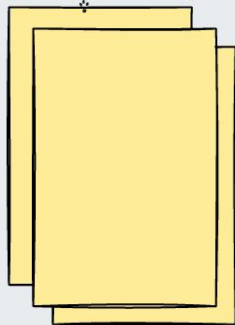
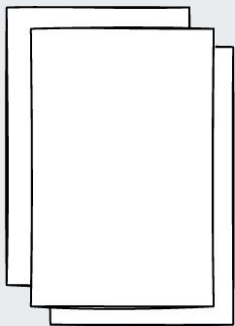
Heap Pages

Vacuum Process removes Page



Heap Pages

Latest page inserted anywhere it fits



How to make a Postgres

WAL

Heap Pages + MVCC (Multi Version Concurrency Control)

B-Tree Indexes

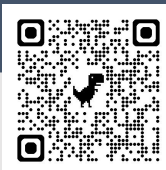


Postgres/MySQL/etc.

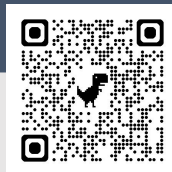
Optimized for updates/upserts and row-level reads

Strong ACID guarantees

Scaling horizontally is challenging



Analytics & Search Architecture



Immutable Parts / Segments

w/ Background Compaction

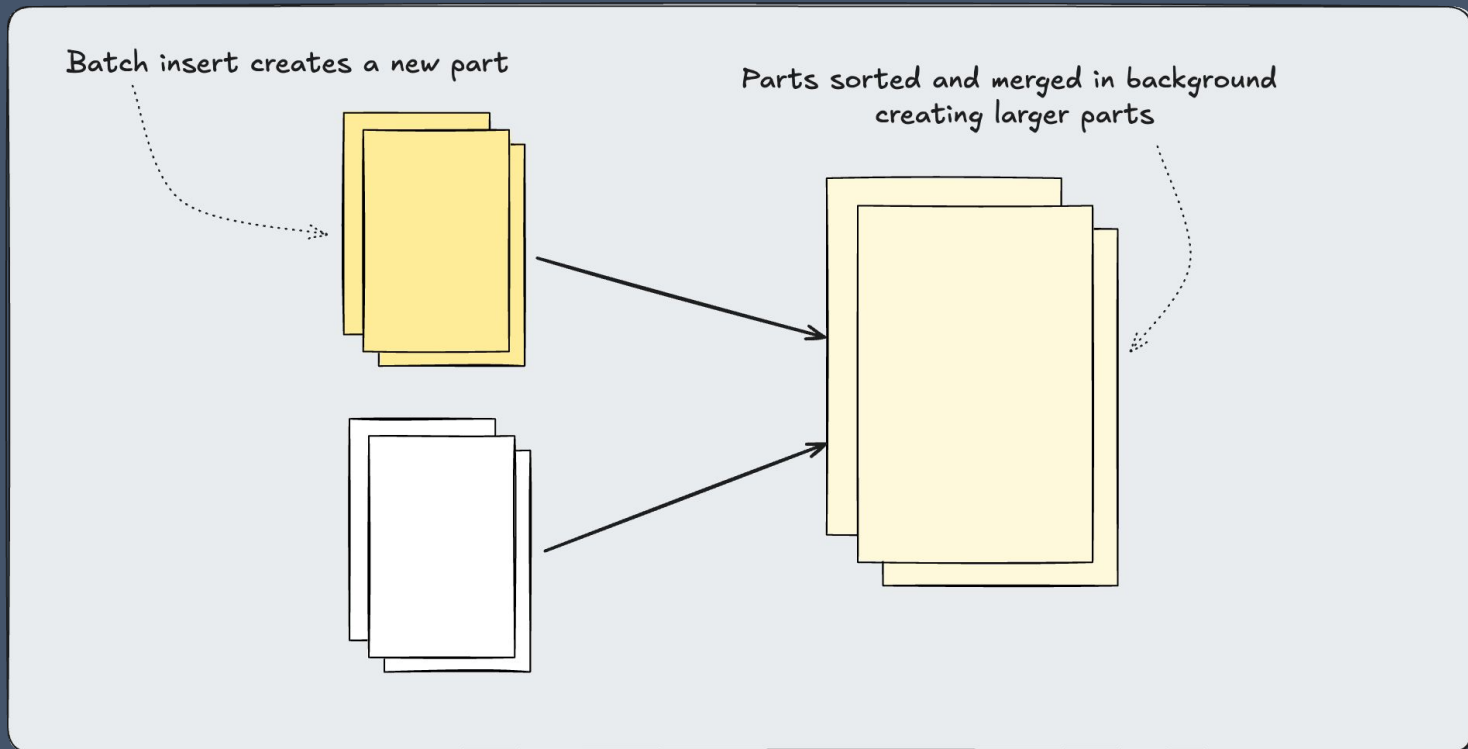


Immutable segment storage:

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



Immutable Parts / Segments



How an LSM Tree Organizes Writes

incoming writes

(k3, v3)

(k1, v1)

(k4, v4)

(k2, v2)

sorted in memory

k1	v1
k2	v2
k3	v3
k4	v4

writes → memory →
immutable runs → compaction

flush

Level 0

run 0

k1	v1
k2	v2

run 1

k2	v2
k3	v3

run 2

k3	v3
k4	v4

run 3

k5	v5
k6	v6

background
compaction

Level 1

run A

k1	v1
k2	v2
k3	v3

run B

k4	v4
k5	v5
k6	v6

Search & Vector Engines

Inverted Indexes

Bloom Filters

Approximate Nearest Neighbor (ANN Graph)



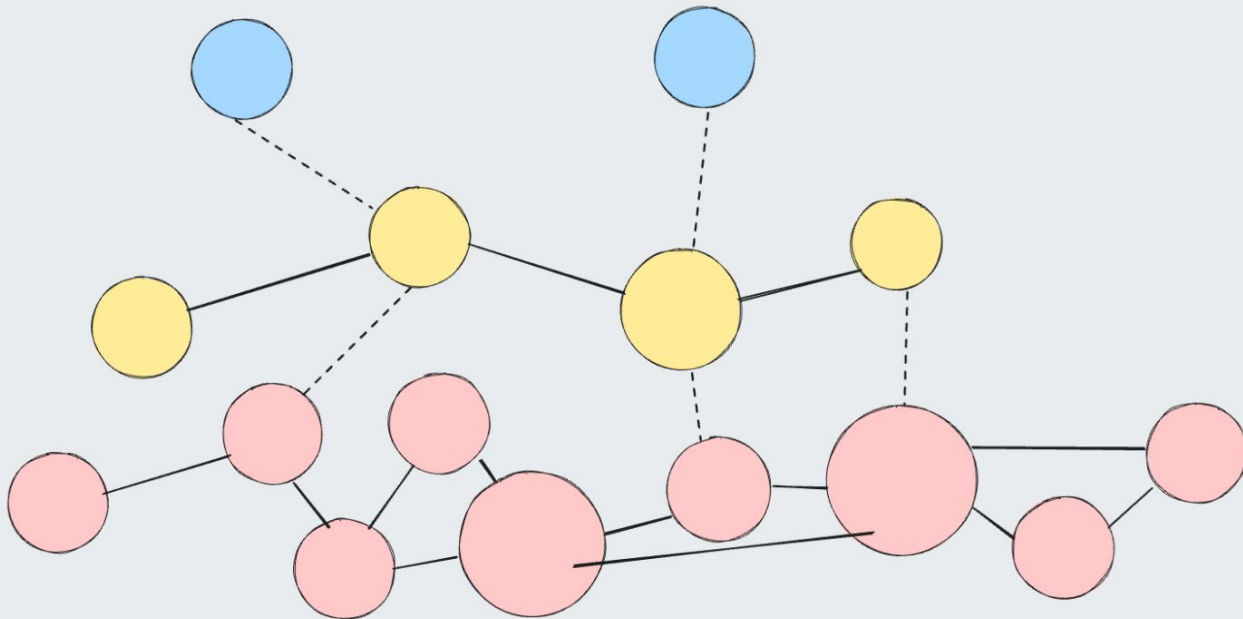
Inverted Indexes

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

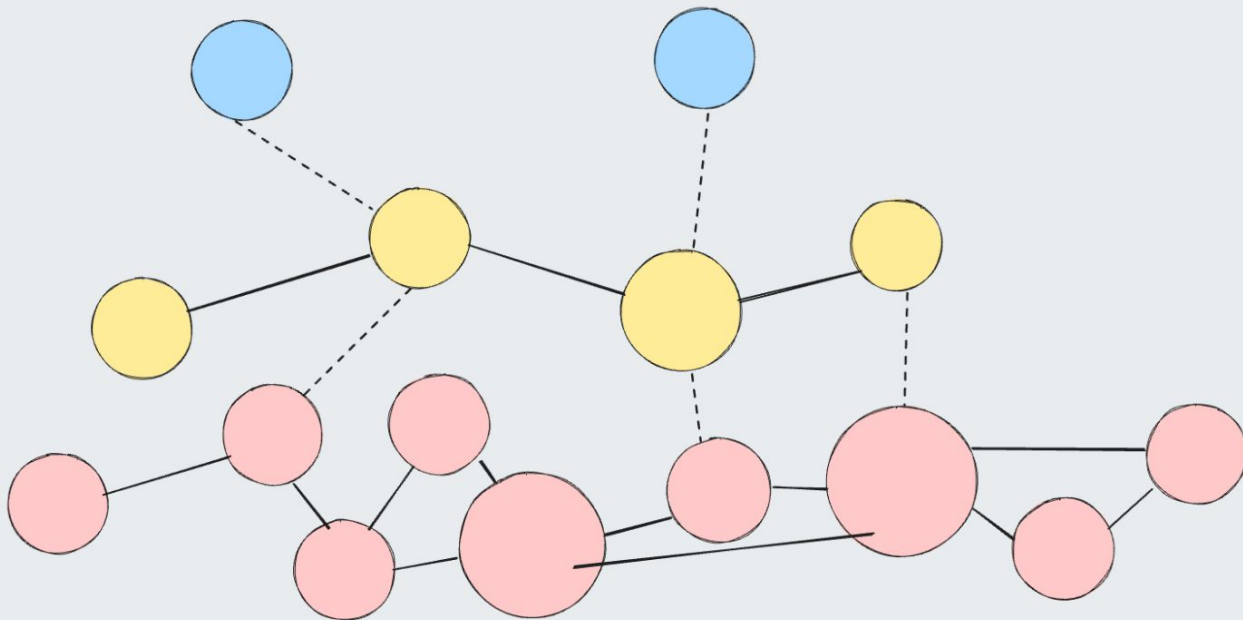


Approximate Nearest Neighbor (ANN)

A way to organize and filter vectors



Approximate Nearest Neighbor (ANN) (Hierarchical Navigable Small World)



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

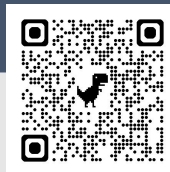
```
"otel-collector-prod-01" → 31  
"otel-collector-prod-10" → 31
```

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



Prometheus (& friends)

Time-series Database



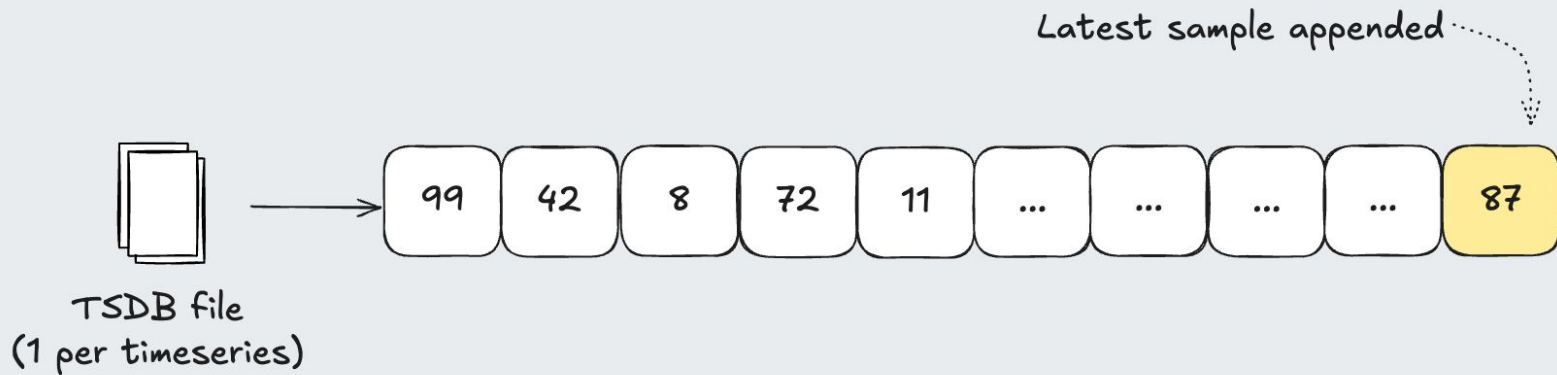
TSDB Blocks

Append-only



TSDB: Data is naturally ordered by time

Excellent for frequent reads of last-sample



TSDB

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

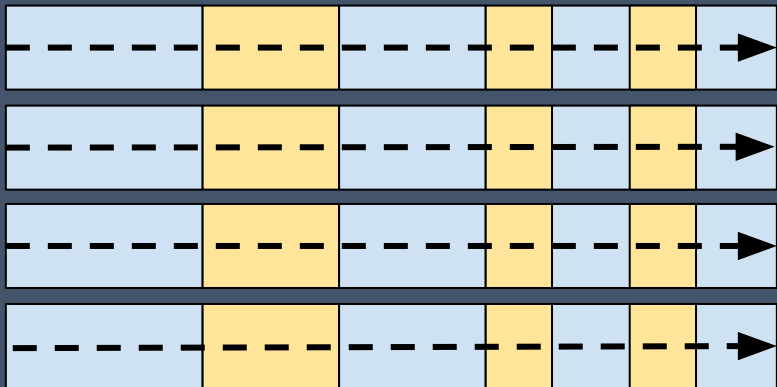


Row-oriented vs column-oriented storage



Row-oriented Storage

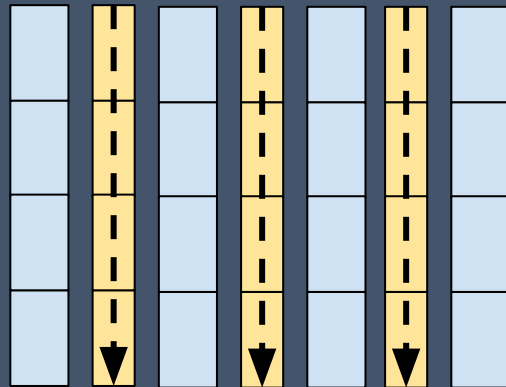
Read all columns in row



Rows compressed minimally or not at all

Column-oriented

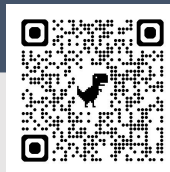
Read only selected columns



Columns highly compressed

ClickHouse

Column-oriented MergeTree



59 GB
(100%)

Read 109
columns

Read 3 columns
from 109

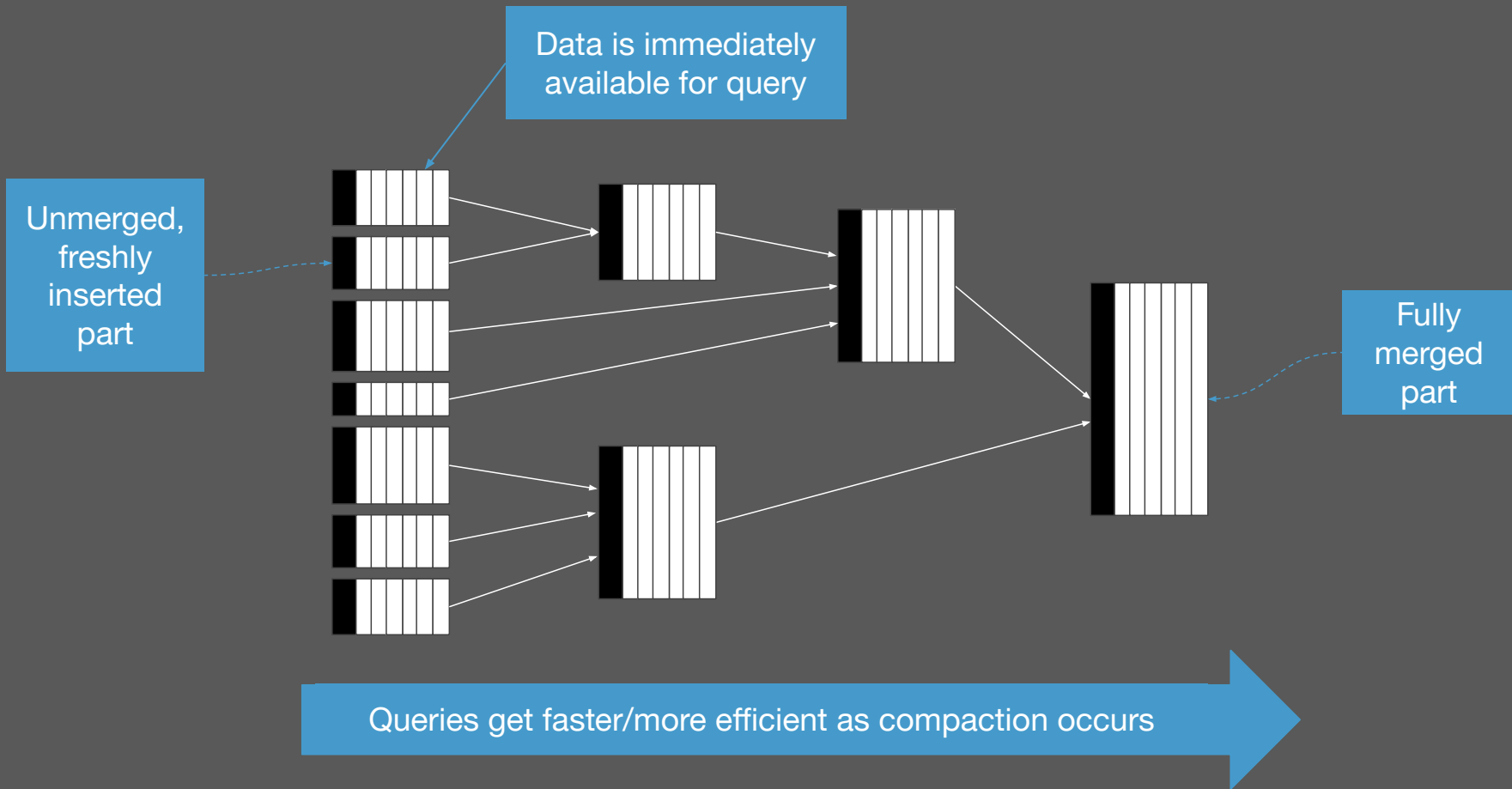
1.7 GB
(3%)

Read 3
compressed
columns

21 MB (.035%)

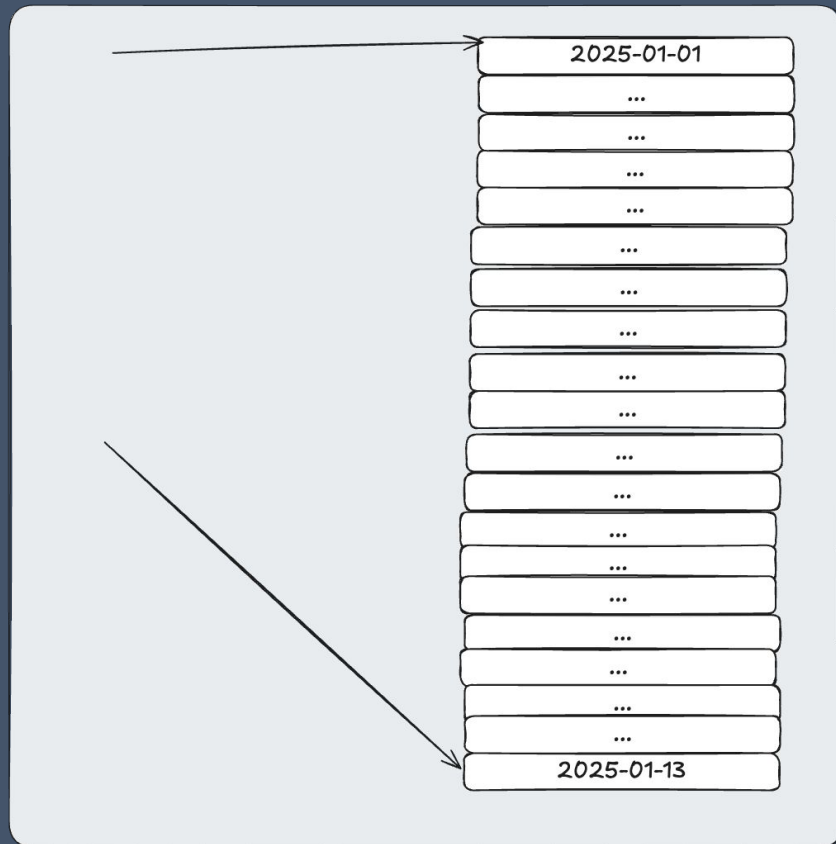
Read 3
compressed
columns over 8
threads

2.6 MB
(.0044%)



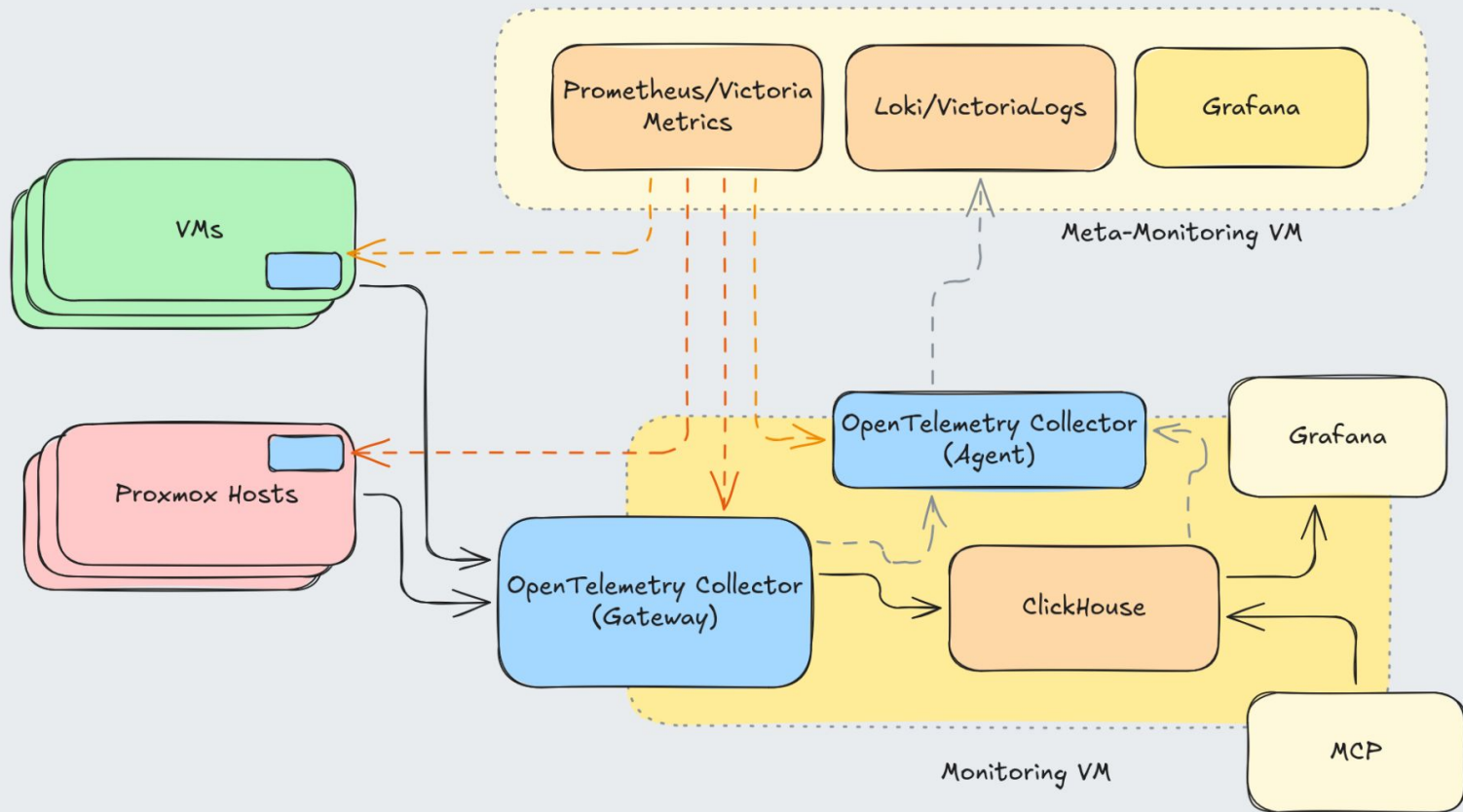
Sparse Indexes

Quickly & cheaply find data
based on an ordered key



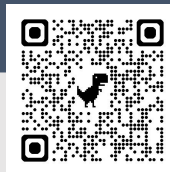
```
graph LR; OT[OpenTelemetry Collector] --> CH[ClickHouse]; OT --> P[Prometheus]; OT --> OS[OpenSearch];
```

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. Arrows show the flow from the sources to the collector and then to the sinks. On the right, three cyan rounded rectangles represent the sinks: ClickHouse, Prometheus, and OpenSearch. Arrows show the flow from the collector to these sinks. Specifically, OpenTelemetry Traces feeds into the Traces Pipeline, which then feeds into ClickHouse. SNMP Metrics, Prometheus Metrics, and Container Logs feed into the Metrics Pipeline, which then feeds into Prometheus. Kubelet Logs feeds into the Logs Pipeline, which then feeds into OpenSearch. There is also a dashed arrow from the Traces Pipeline to the Metrics Pipeline.



VictoriaMetrics

TSDB meets MergeTree



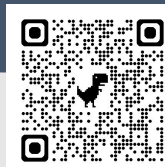
Loki

Uncomplicated logging (with label indexes)



Honorable Mentions

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



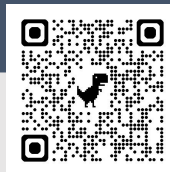
At (very) small scale

Just use what you have until it breaks (Postgres)



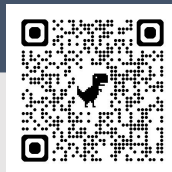
Hooked-on full-text search

Elastic/OpenSearch has your back



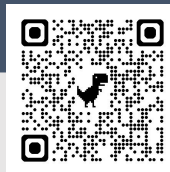
One database for everything

ClickHouse is pretty cool



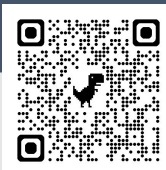
Wide-event analytics

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 (Columnar)	OLTP/CQL	Segments	Inverted, Bloom Filters, ANN	Scalable upserts
Prometheus (Columnar*)	TSDB/PromQL	TSDB files	By label	Time-series metrics, alerting
OpenSearch (Document)	Search/DSL	Lucene Segments	Inverted, ANN	Full-text search
ClickHouse (Columnar)	OLAP/SQL	MergeTree Parts	Sparse, Inverted, and more...	Real-time analytics



Happy querying!

Download slides, connect with me →

