A Roundup of Observability Datastores

Josh Lee • Altinity • OSOD • Oct. 23, 2025

"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?

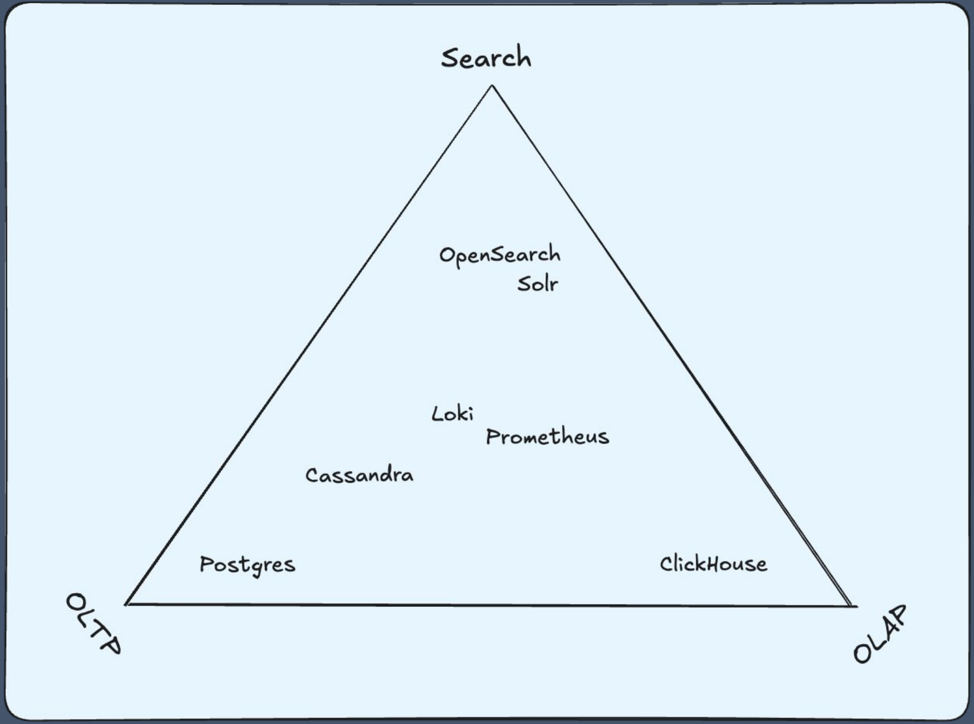
Database Archetypes

OLTP
OLAP
TSDB
Search/Analytics

Introducing the cast of characters

Postgres (OLTP)
Cassandra (OLTP)
OpenSearch (Search & Analytics)
Prometheus (TSDB)
ClickHouse (OLAP)

Taxonomies are challenging



Storage on disk

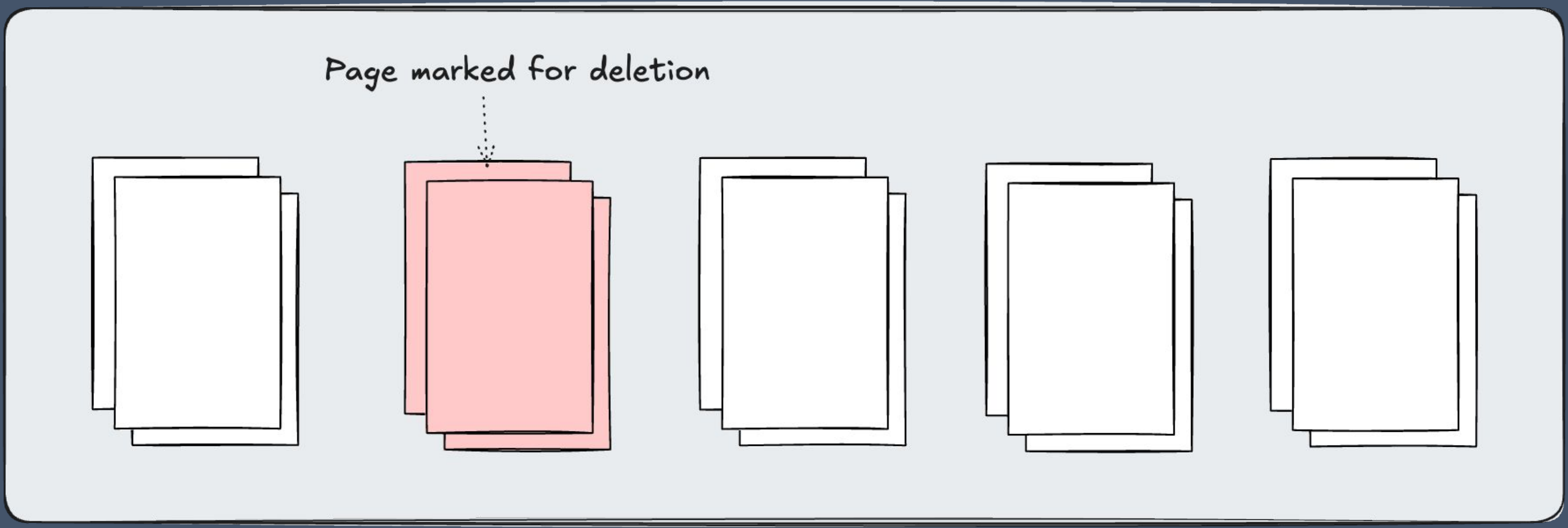
Database Storage Styles

Heap Pages + Commit Log
Time-series Blocks
Parts / Segments

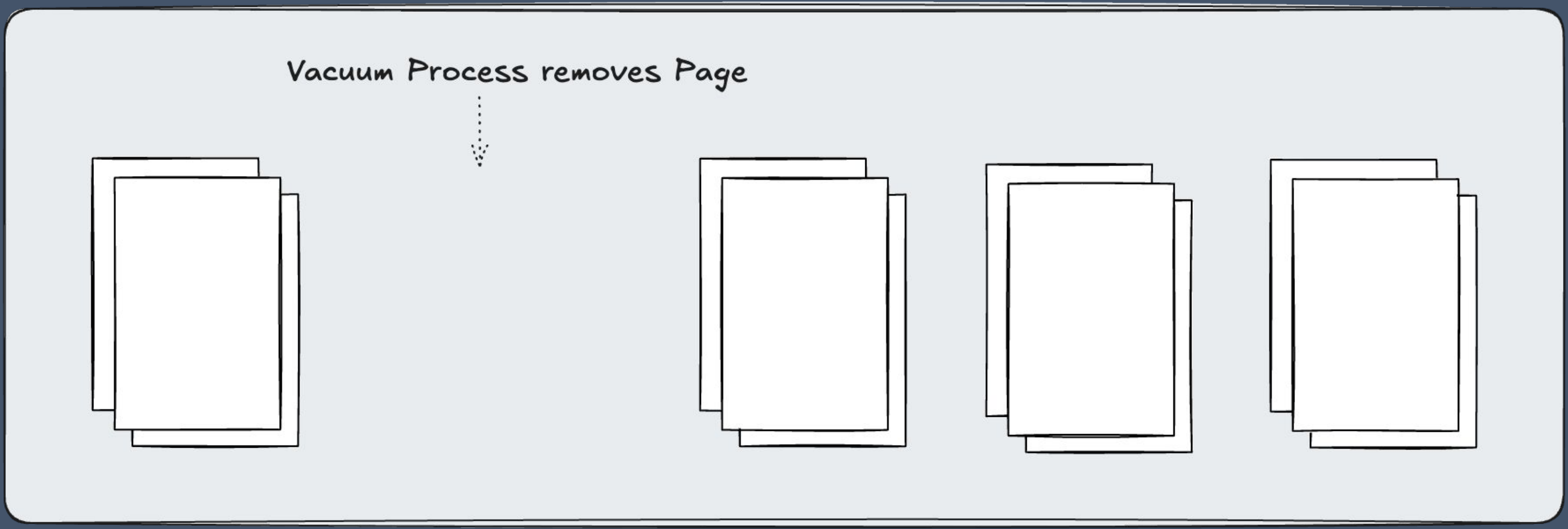
Heap Pages

* the JBOD of storage styles

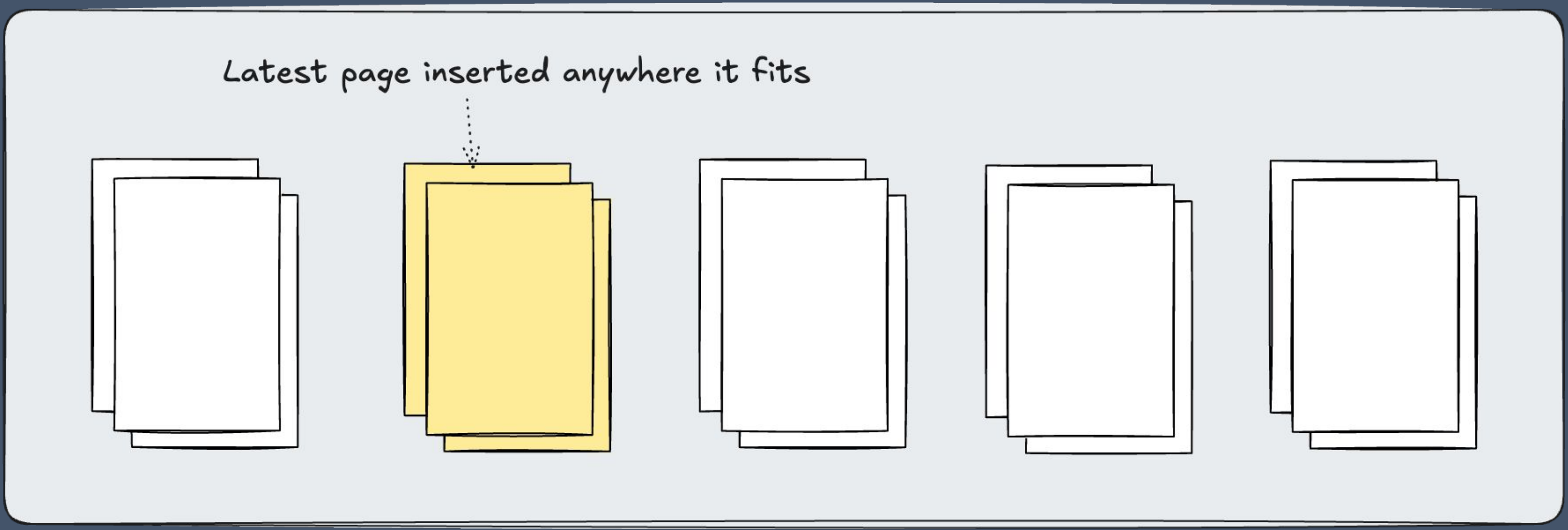
Heap Pages



Heap Pages



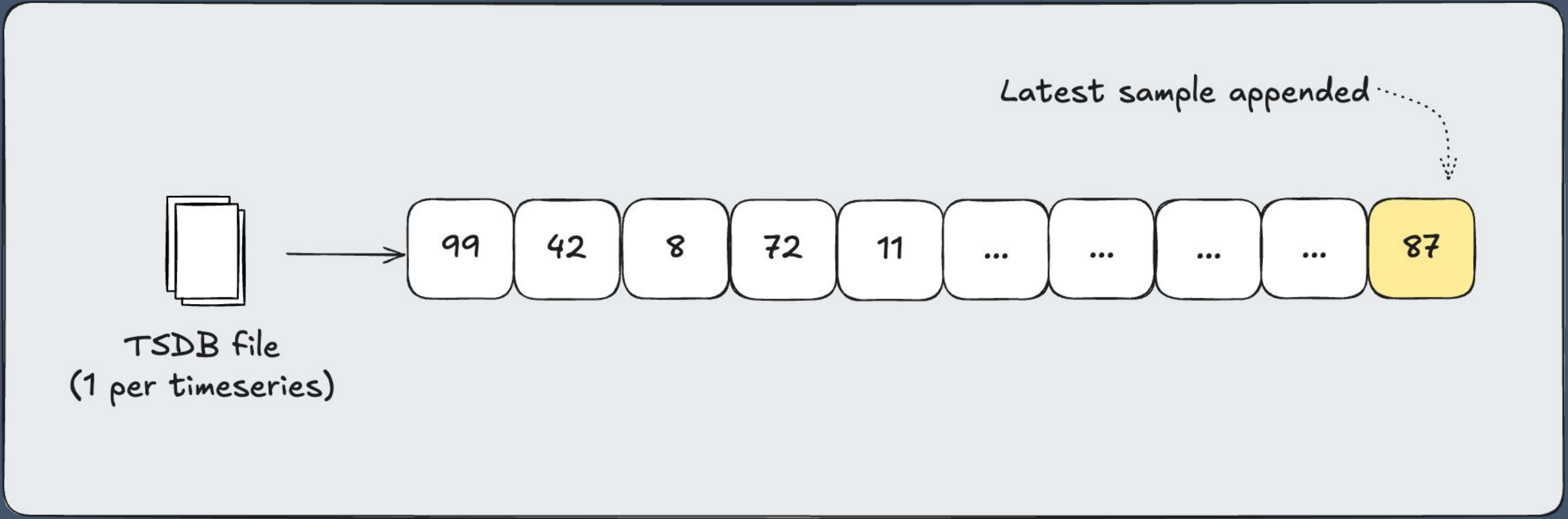
Heap Pages



TSDB Blocks

Append-only

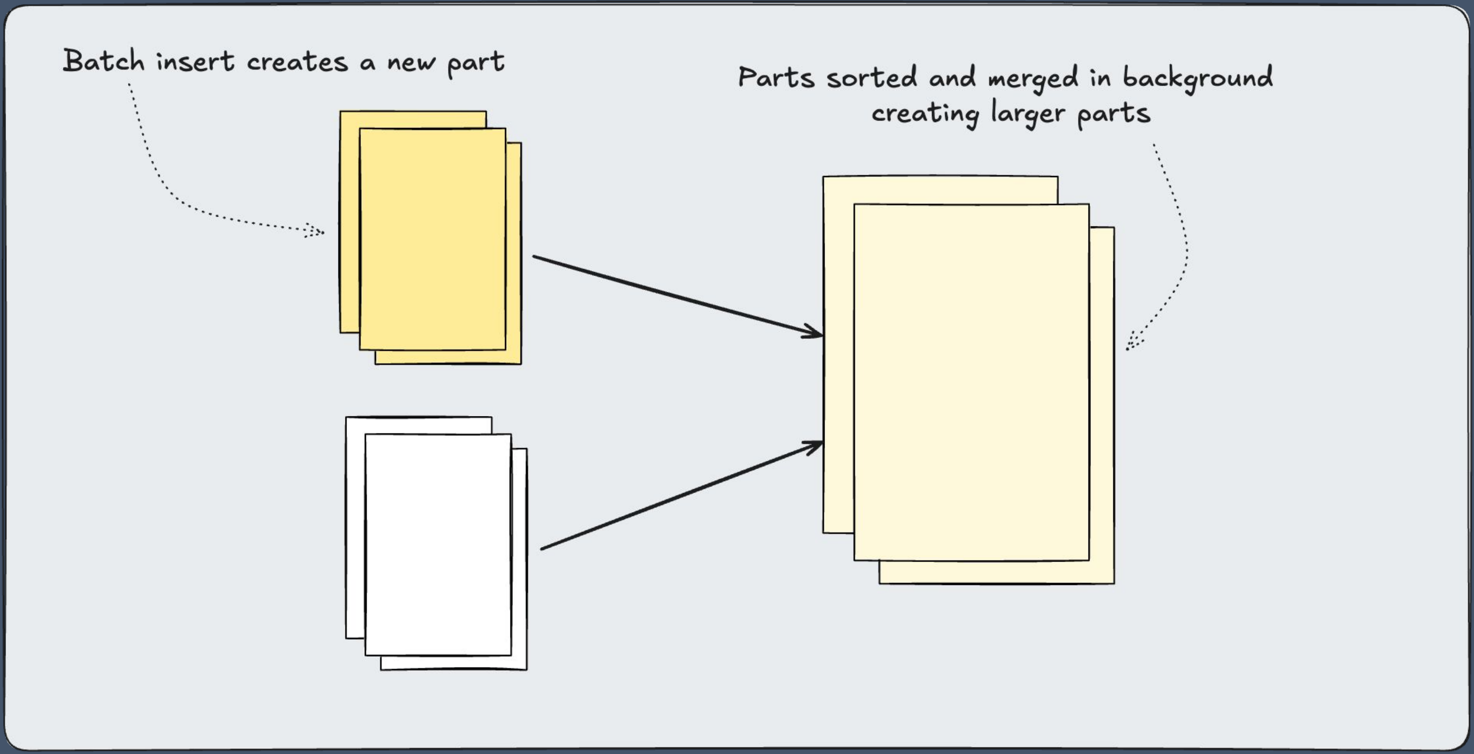
TSDB Blocks


Latest sample appended
99
42
8
72
11
...
...
...
...
87
TSDB file
(1 per timeseries)

Immutable Parts / Segments

w/ Background Compaction

Immutable Parts / Segments



Batch insert creates a new part
Parts sorted and merged in background creating larger parts

Writing Data

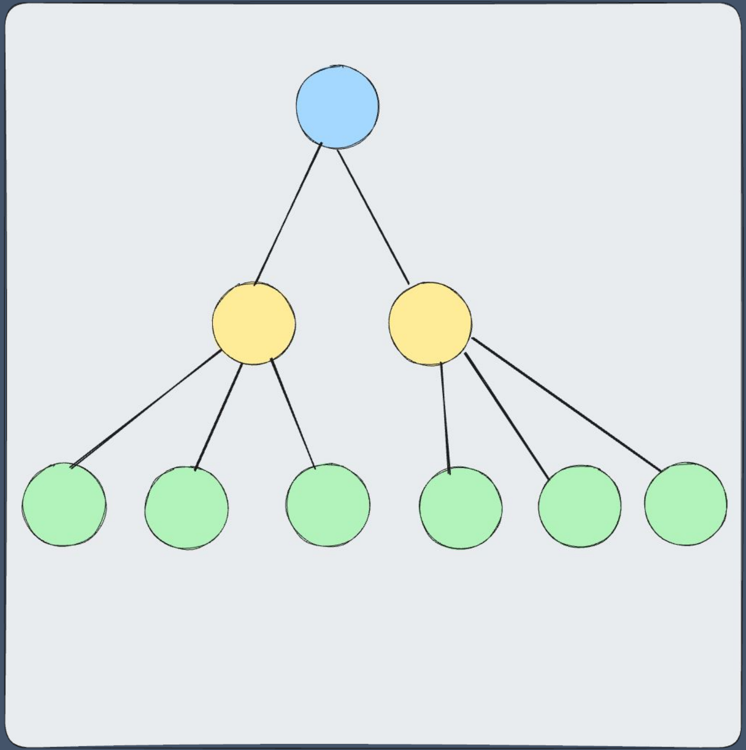
Write Ahead Log (WAL) / Commit Log

Buffered, unordered writes stored on disk

Concurrency Control Strategies

- MVCC + Vacuum
- "Tombstone" deletes
- Last-write wins

Balanced Trees (B-Trees)



Now we can build a Postgres

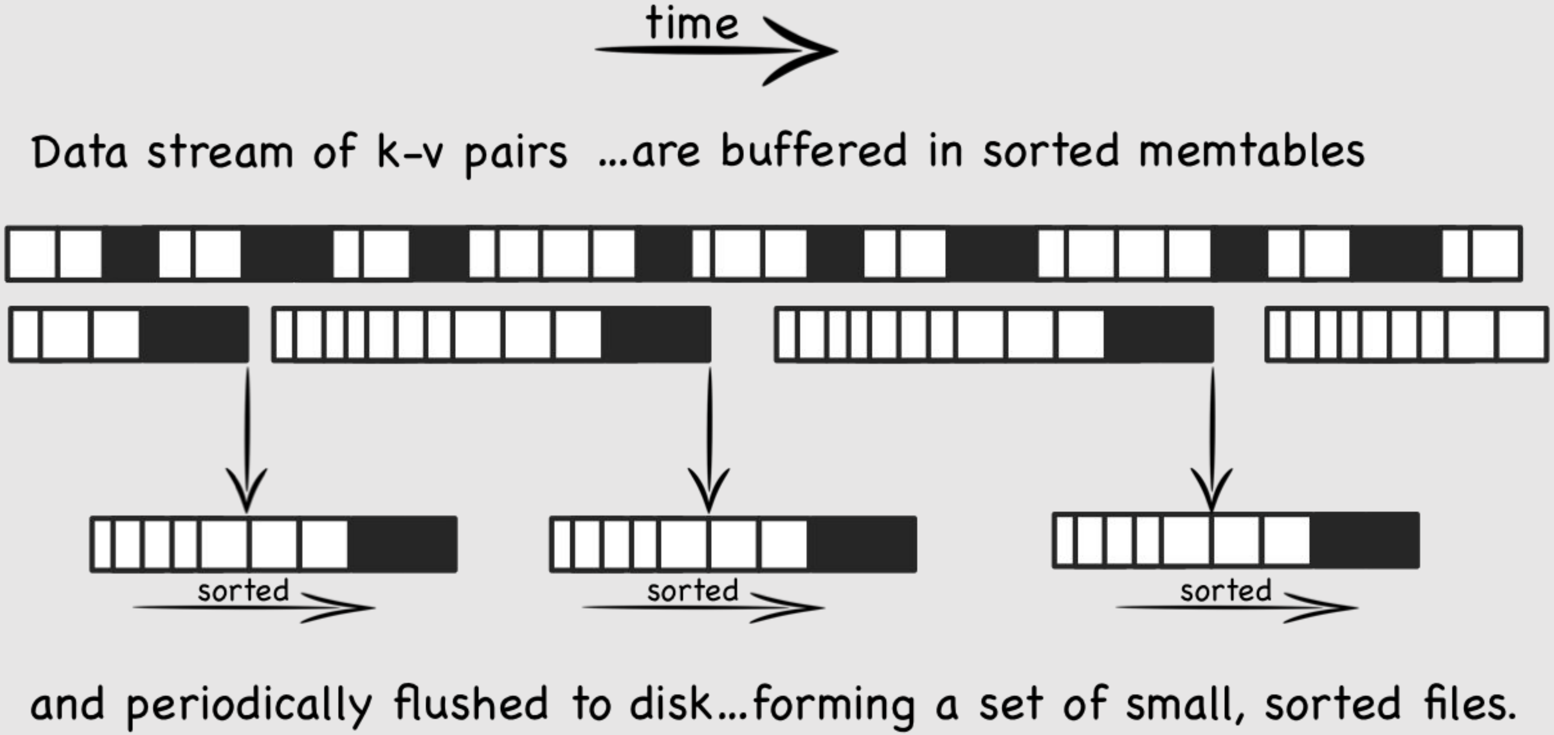
WAL
Heap Pages + MVCC
B-Tree Indexes

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



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

and periodically flushed to disk...forming a set of small, sorted files.

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

Lucene Family:

Cassandra, Elastic/OpenSearch, Apache Solr

A Lucene Query

```
(title:"database systems" OR content:(postgres OR "clickhouse"))
AND timestamp:[2025-01-01 TO 2025-12-31]
AND NOT tags:deprecated
```

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]
```

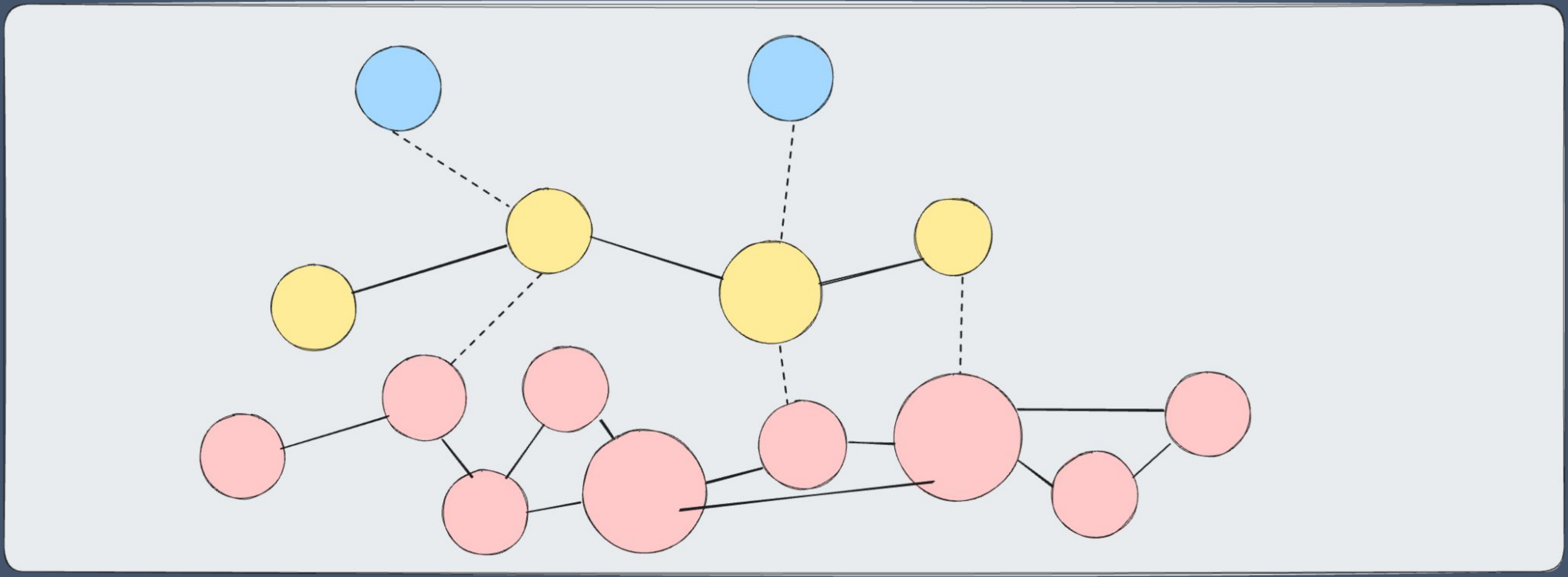
Bloom Filters

No false negatives

```
"ae7c" → [doc1, doc3, doc7]
"f9b2" → [doc2, doc5]
"8c93" → [doc1, doc4]
```

Approximate Nearest Neighbor (ANN)

A way to organize and filter vectors

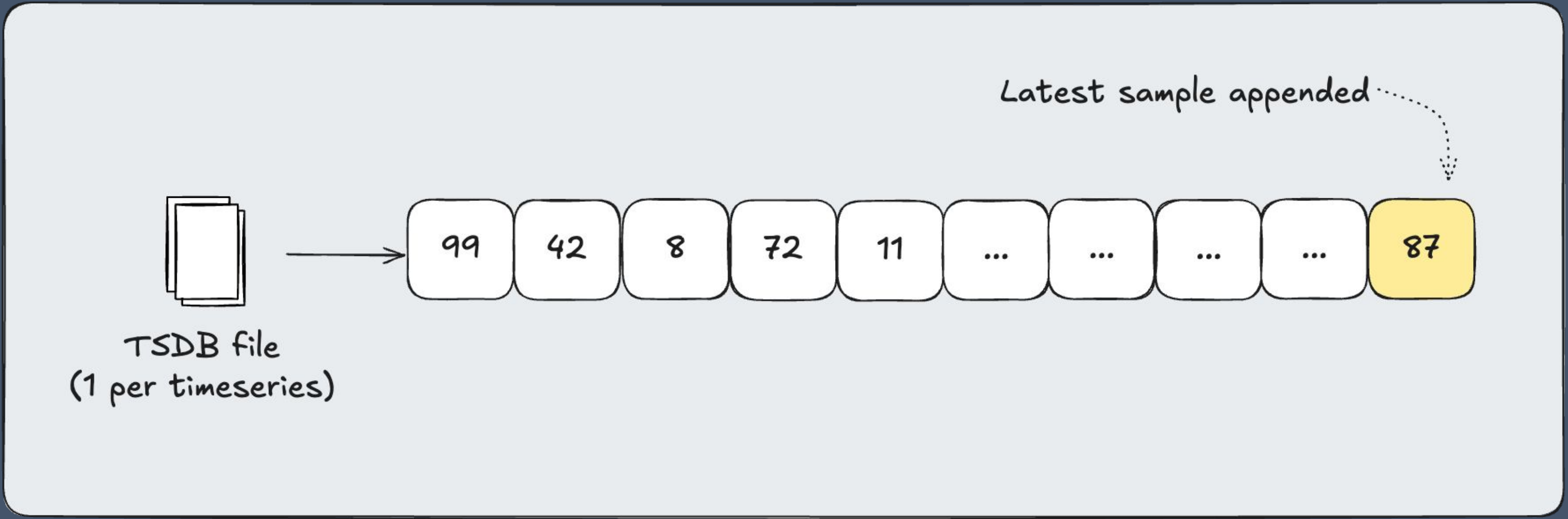


Prometheus (& friends)

Time-series Database

TSDB: Data is naturally ordered by time

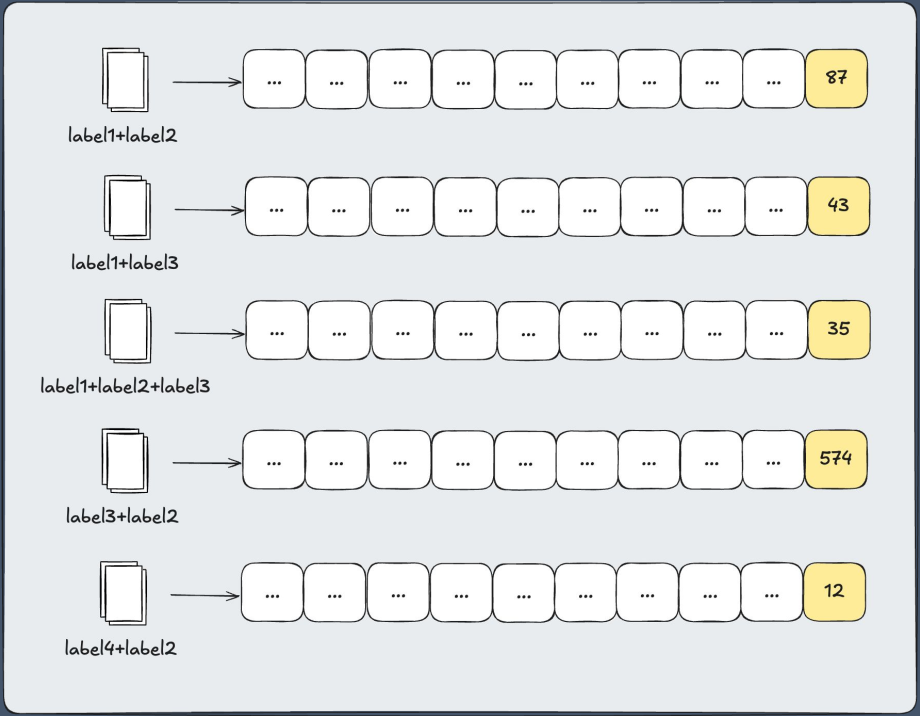
Excellent for frequent reads of last-sample



TSDB

No. of time series = cardinality^dimensionality

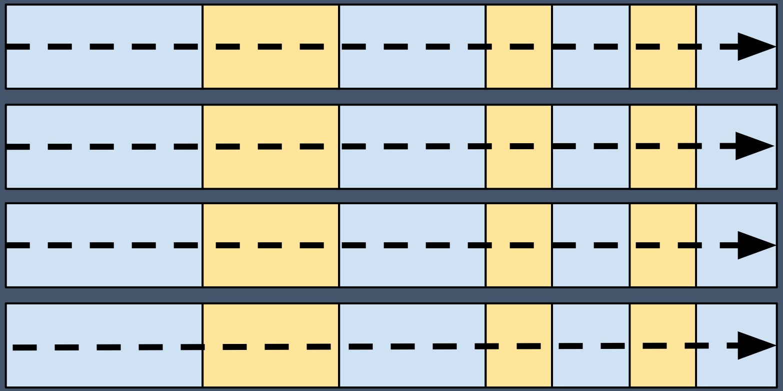
Cardinality Explosions



Row-oriented vs column-oriented storage

Row-oriented Storage

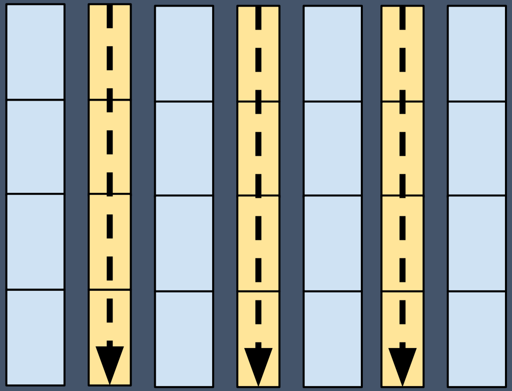
Read all columns in row



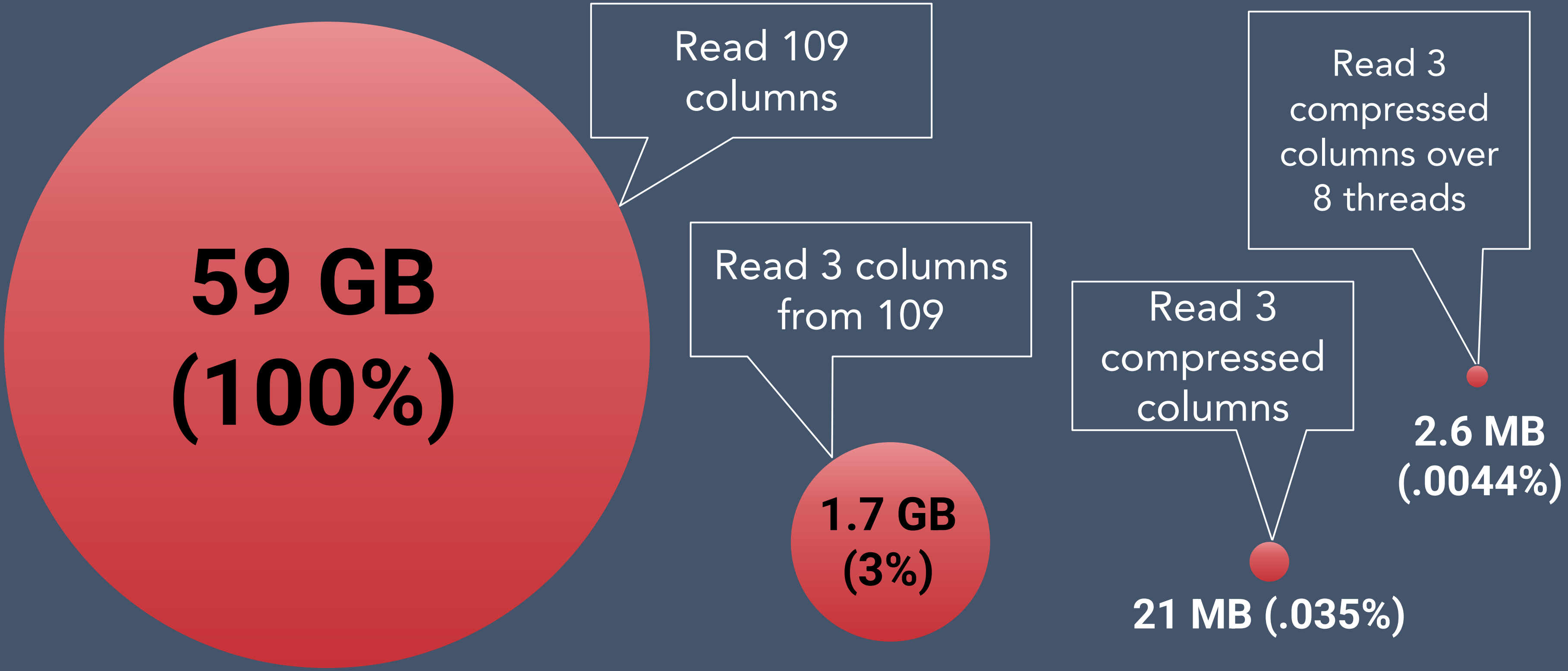
Rows compressed minimally or not at all

Column-oriented

Read only selected columns



Columns highly compressed



59 GB (100%)

Read 109 columns

1.7 GB (3%)

Read 3 columns from 109

21 MB (.035%)

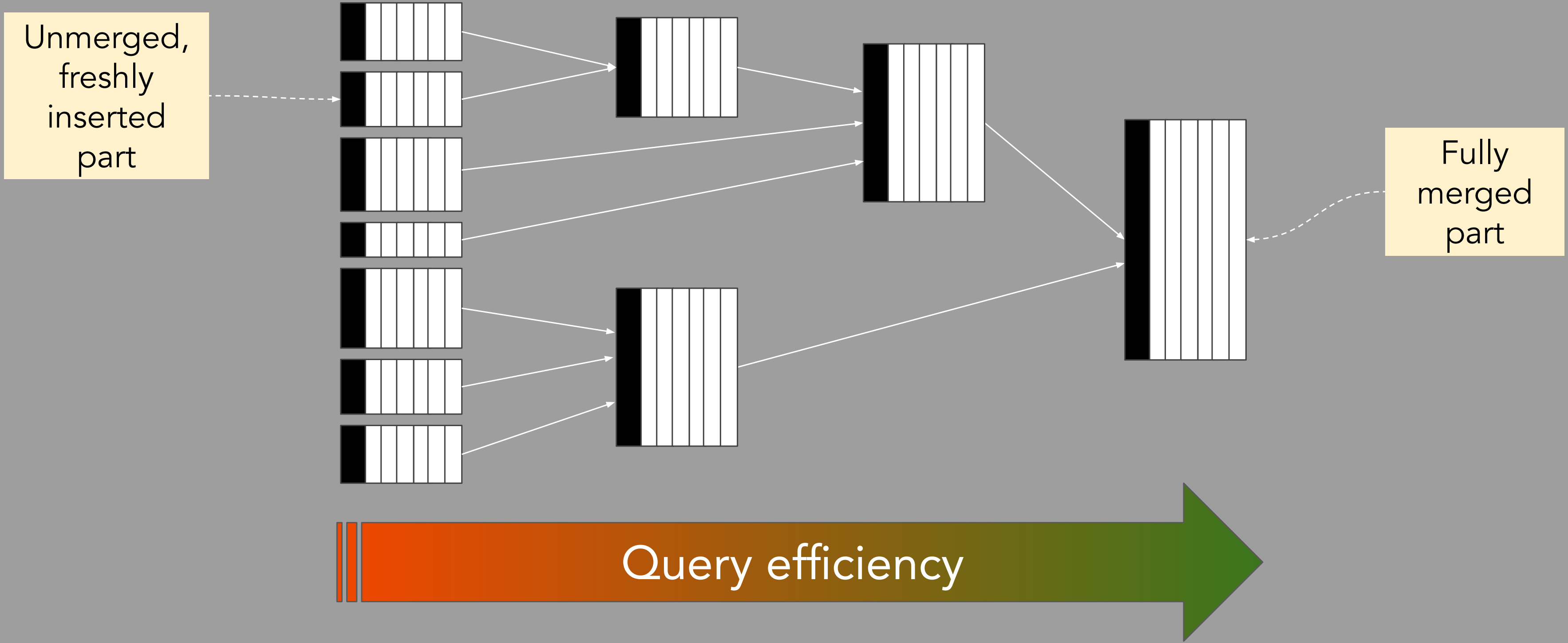
Read 3 compressed columns

2.6 MB (.0044%)

Read 3 compressed columns over 8 threads

ClickHouse

Column-oriented MergeTree



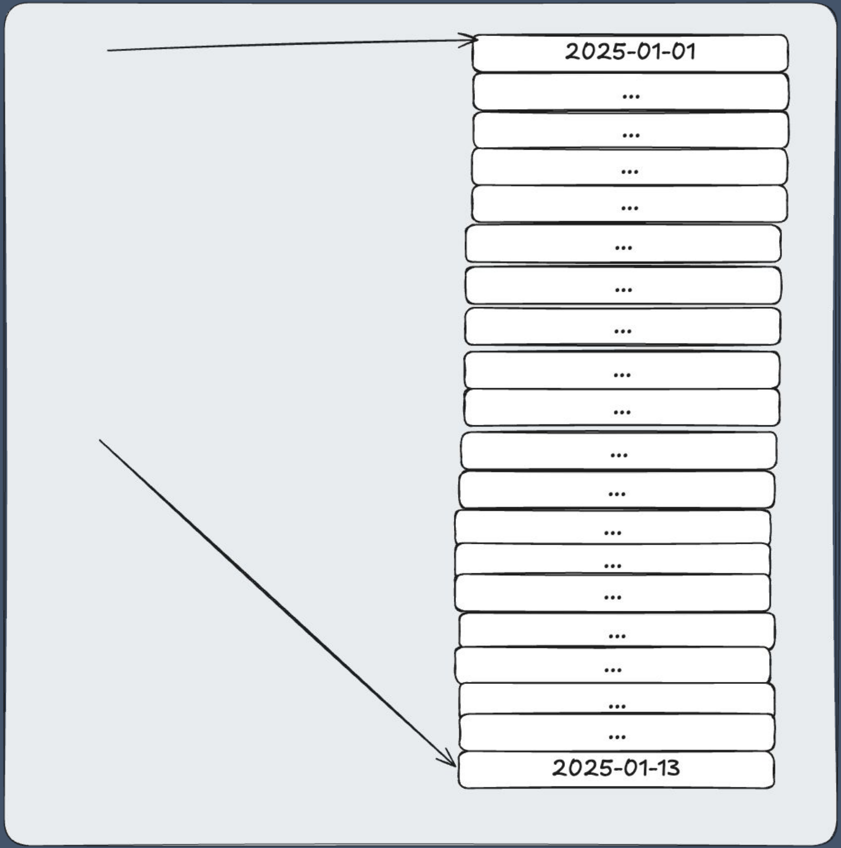
Unmerged, freshly inserted part

Fully merged part

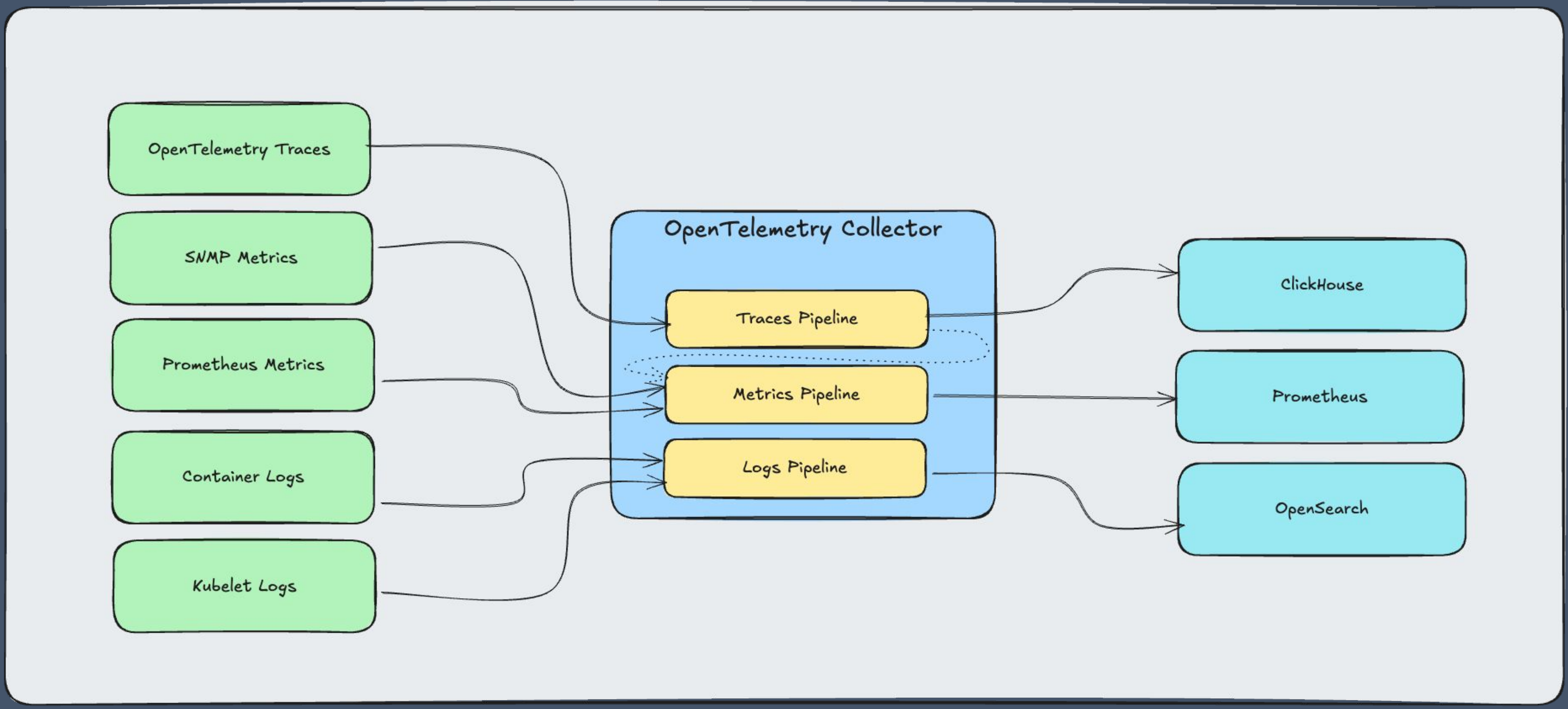
Query efficiency

Sparse Indexes

Quickly & cheaply find data based on an ordered key



Which to choose?



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

OpenSearch has your back

One database for everything

ClickHouse is pretty cool

Wide-event analytics

ClickHouse is awesome

Filtering heavily before analyzing

OpenSearch is also a good choice here

Lots of "last-sample" reads + alerts

Choose a TSDB like Prometheus or VictoriaMetrics

Wide-events analytics with transactional guarantees

Cassandra or Postgres->ClickHouse

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, analytics
ClickHouse (Columnar)	OLAP/SQL	MergeTree Parts	Sparse, Inverted, and more…	Wide-event analytics

Thank you and happy querying!

Josh Lee - Altinity