TimescaleDB
BY TIGER DATA

74%

of IIoT pilots **never reach production.**

The database is usually where they die.

The pilot lies to you.

PoC

millions of rows

Inserts: **milliseconds**

Dashboards: **instant**

Cost: **rounding error**

PRODUCTION

billions of rows

Inserts: **backlogs forming**

Dashboards: **loading…**

Cost: **someone notices**

Different physics. Same database.

IIoT data is multiplicative.

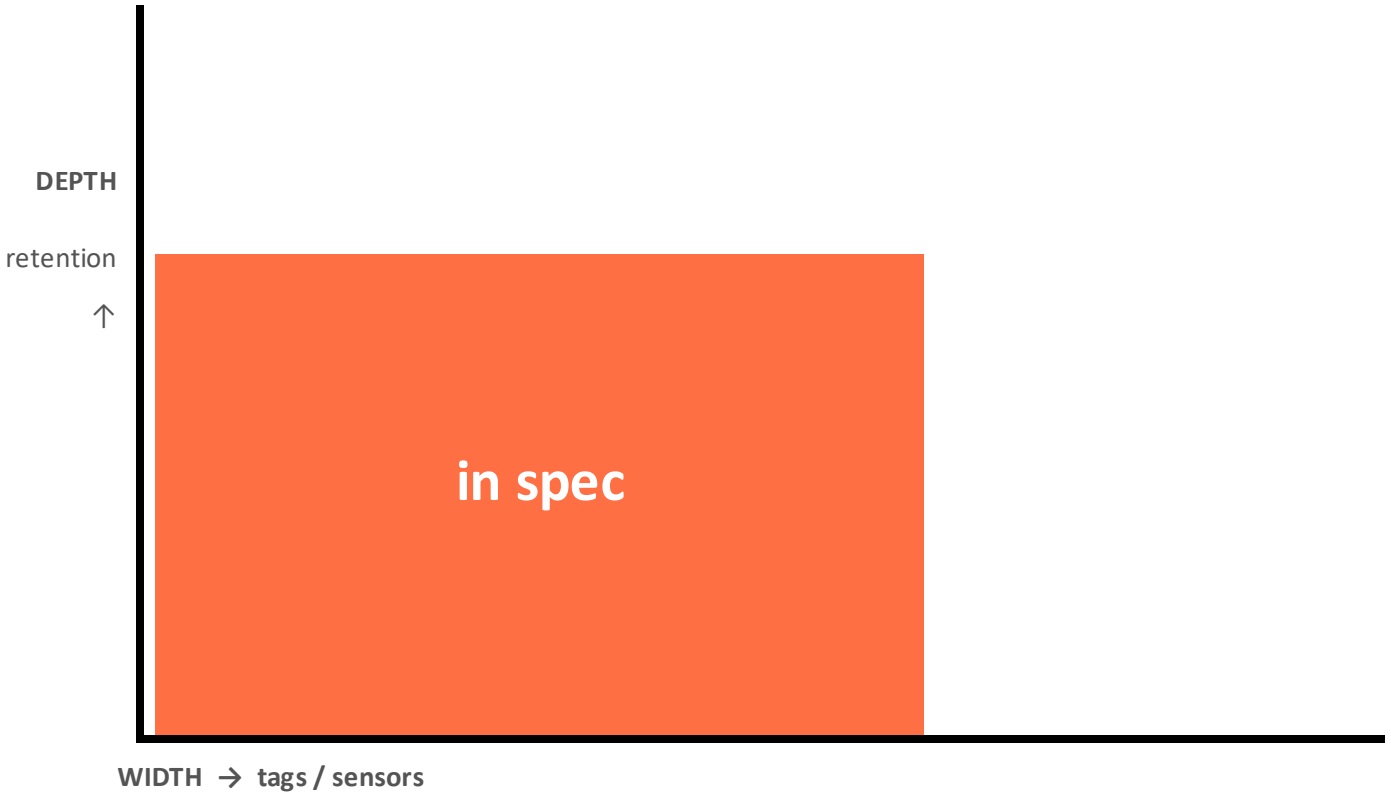
volume = **n** × **v** × **r** × **t**

tags data/tag frequency time

ONE SMALL FACTORY / 10,000 tags × 1 Hz × 1 year

315 billion rows • 37.5 TB on disk

Every IIoT system has an envelope.



Three walls define the box:

01 STORAGE

cost compounds with retention

02 INGEST

throughput collapses without warning

03 QUERY

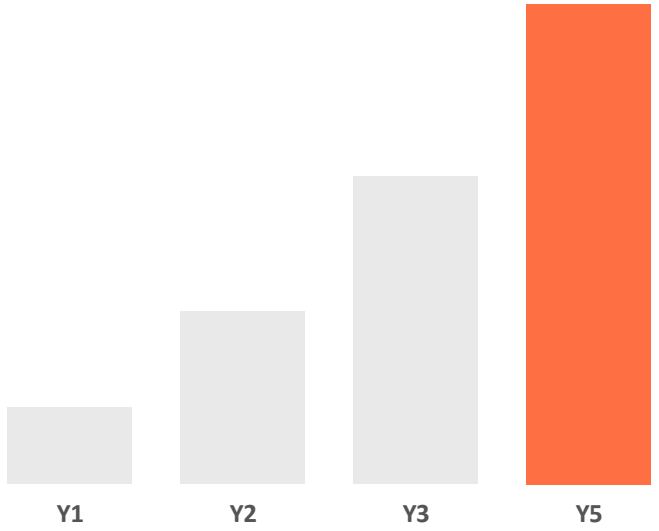
aggregates slow with data growth

Storage compounds quadratically.

cumulative cost $\propto n \times r \times T^2$

Trivial in month 1.

Exorbitant in year 5.



cumulative storage cost

Source: Pagnutti, tigerdata.com/blog/the-iiot-postgresql-performance-envelope

Ingest fails without warning.

WAL writes and index updates **grow with DB size.**

System looks healthy for months.

Then hits equilibrium with no runway.

Backlog forms. Backlog grows.

You don't catch up.

THE RULE

<80%

of max ingest.

One bad VACUUM, one network blip, one burst, and you're behind forever.

Queries degrade linearly.

Indexed deep queries: **O(log t·r)**

Aggregate dashboards: **O(t·r)** linear with retention.

Every dashboard you've ever shipped uses **GROUP BY** and **AVG**.

They scan every row in the window. The window keeps growing.

PLUS THE SLOW ROT

- Planner stats go stale
- VACUUM falls behind
- Server hardware drifts
- Sequential scans creep back in

DASHBOARD LOAD TIME

milliseconds → **seconds** → **minutes** → **loading…**

Hardware buys time. It doesn't move the ceiling.

01 VERTICAL SCALING

Bigger box.

More RAM, CPU, disk.

Diminishing returns.
Hard ceilings.
Doubling RAM never doubles anything that matters.

02 HORIZONTAL / SHARDING

More boxes.

Split tables, shard the DB.

Replication lag.
Cross-node coordination.
Often breaks the ACID guarantees you chose PG for.

03 ADD A SECOND DB

InfluxDB. A historian.

Leave Postgres for analytics.

Pipelines. Sync. Drift.
Two systems, one team.
A permanent operational tax.
New query language to learn.

All three push the wall out. None of them change the math.

Change the math. Stay on Postgres.

WALL 02 → INGEST

Hypertables

Auto-partition by time. Indexes stay in memory per chunk.

PG cliff at ~25M rows.
Hypertables stay flat.

WALL 03 → QUERY

Continuous aggregates

Dashboards read precomputed rollups, not raw rows.

250-1500 ms → 0.4 ms

WALL 01 → STORAGE

Hypercore (columnar)

Older chunks compress columnar. Same SQL, less disk.

80-95% compression.
$150K/yr → ~$15K.

Benchmarks: tigerdata.com/blog/how-timescaledb-expands-postgresql-iiot-performance-envelope

All three primitives ship as a Postgres extension. Same SQL. Same tooling. No second database.

06 / THE OUTCOME

AXPO / ENERGY UTILITY

"We connected 2 power plants.
After half a year the database was completely full,
we couldn't insert or query any data,
and the whole system crashed."

Emanuel Joos, Lead Software Engineer, Axpo

Now: **20+ systems. Still running.**

Map your envelope. tigerdata.com/blog/the-iiot-postgresql-performance-envelope

WHO'S TALKING

Matty Stratton

Sign up for a trial and get $1000 in credits!

https://tsdb.co/matty-iot-tech

Head of Developer Advocacy & Docs at Tiger Data and founder of the Arrested DevOps podcast. I live in Chicago with too many pets.

Passionate advocate for helping developers make better database decisions and get more out of Postgres and time-series data

linkedin.com/in/mattstratton/ (slides at speaking.mattstratton.com)