pg-flight-recorder

A Black Box for PostgreSQL

David A. Ventimiglia

Supabase

The Problem

> "What was happening in my database?"

- 3 AM: Users complaining, dashboards red
- You wake up, check PostgreSQL... everything looks fine now
- **The incident ended. The evidence vanished.**

PostgreSQL forgets the past:

- `pg_stat_activity` shows **now**, not **then**
- No built-in history of wait events, locks, or queries

The Solution

Flight recorder concept: Always recording, ready when you need it

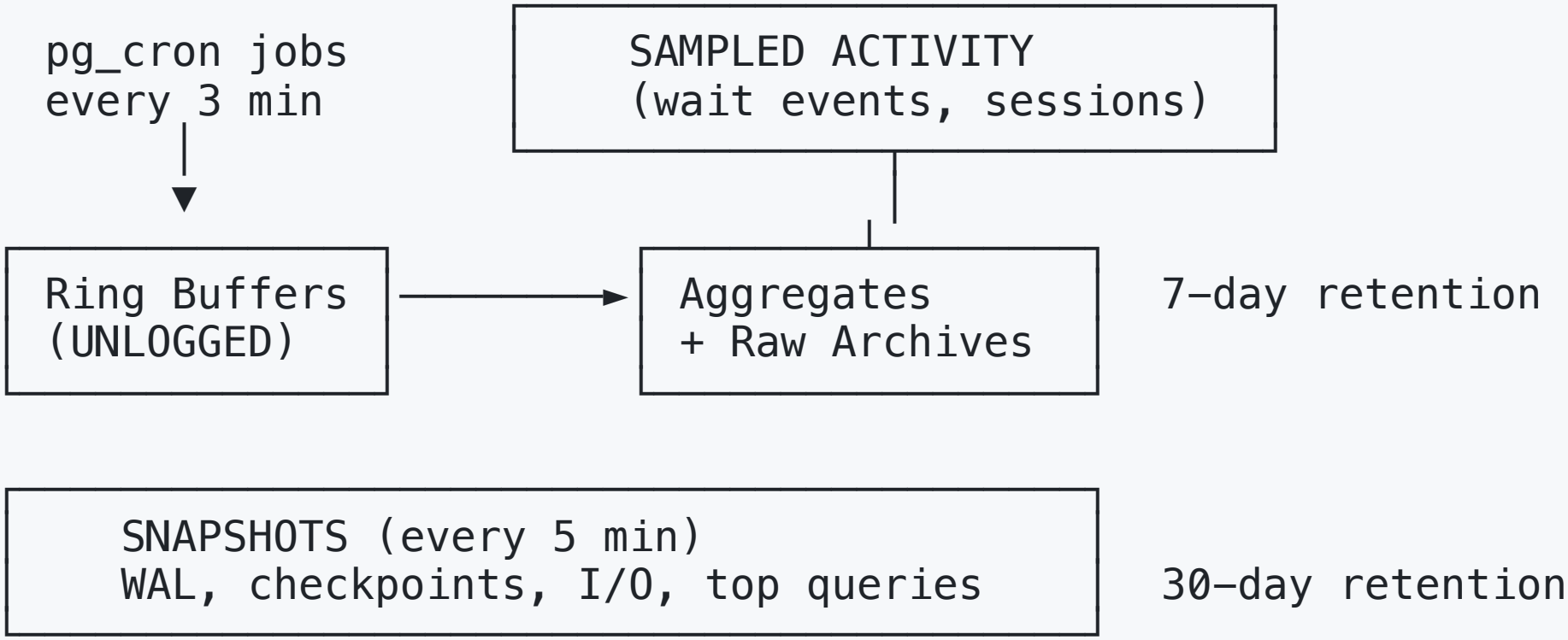
Like an airplane's black box - you don't think about it until you need it

- **Server-side** - runs inside PostgreSQL via pg_cron
- **Zero-config** - `psql -f install.sql` and done
- **Production-safe** - battle-tested at Supabase scale

What It Captures

Category	Metrics
Wait Events	What backends are waiting on
Active Queries	Running queries with session age
Locks	Lock contention and blocking chains
WAL & Checkpoints	Write-ahead log activity
I/O Timing	Storage latency by backend type
Top Queries	Slowest/most frequent queries
Table/Index Stats	Hotspots, bloat, unused indexes
Config Changes	Parameter drift detection

How It Works


pg_cron jobs
every 3 min
SAMPLED ACTIVITY
(wait events, sessions)
Ring Buffers
(UNLOGGED)
Aggregates
+ Raw Archives
7-day retention
SNAPSHOTS (every 5 min)
WAL, checkpoints, I/O, top queries
30-day retention

Production Safe

Measured overhead: ~0.02% CPU

Metric	Value
Median collection	23ms
P95	31ms
Collections/day	480

Built-in protection:

- Load shedding at 70% connection utilization
- Circuit breaker for slow collections
- Auto-disable at 10GB storage
- 0% DDL blocking (tested: 202 operations)

Quick Demo: The Report

```
SELECT flight_recorder.report('1 hour');
```

Output includes:

- Anomalies detected (checkpoints, buffer pressure, locks)
- Wait event summary
- Lock contention events
- Long-running transactions
- Configuration changes
- Table hotspots & index efficiency

Paste into ChatGPT for instant analysis!

Real Investigation: Compare

```sql
SELECT * FROM flight_recorder.compare(
    '2024-01-15 02:00',  -- before incident
    '2024-01-15 03:00'   -- during incident
);
```

```
             metric                | before | during | delta
-----------------------------------+--------+--------+-------
 checkpoint_occurred               | false  | true   |
 bgw_buffers_backend_delta         | 0      | 12847  | +12847
 temp_bytes_delta                  | 0      | 2.1 GB | +2.1GB
```

Diagnosis: Checkpoint during peak + backends writing directly to disk

Anomaly Detection

Auto-detects 12 issue types:

Anomaly	What It Means
`CHECKPOINT_DURING_WINDOW`	I/O spike from checkpoint
`BUFFER_PRESSURE`	Backends bypassing bgwriter
`LOCK_CONTENTION`	Sessions blocked on locks
`XID_WRAPAROUND_RISK`	Transaction ID approaching limit
`IDLE_IN_TRANSACTION`	Sessions blocking vacuum
`DEAD_TUPLE_ACCUMULATION`	Table bloat building up
`REPLICATION_LAG_GROWING`	Replica falling behind

Installation

```
psql -f install.sql
```

That's it. Collection starts automatically.

Requirements:

- PostgreSQL 15, 16, or 17
- pg_cron extension
- Optional: pg_stat_statements for query analysis

Uninstall:

```
psql -f uninstall.sql
```

Try It Today

github.com/supabase/pg-flight-recorder

```
git clone https://github.com/supabase/pg-flight-recorder
psql -f install.sql
# ... wait for an incident ...
SELECT flight_recorder.report('1 hour');
```

MIT Licensed | Open Source | Production-Proven

Questions?