Open Electricity

PostgreSQL + TimescaleDB to ClickHouse

Lessons from migrating ~1B rows of energy data

Nik Cubrilovic — Open Electricity

About Me

Nik Cubrilovic

- nikcub.me
- nc9
- dir
- nineprimes — nineprimes.com

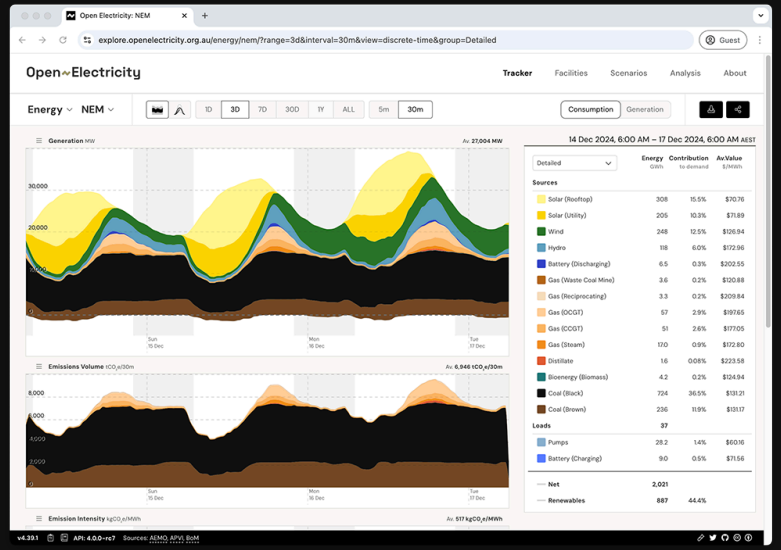
History

- Started as **OpenNEM** at the University of Melbourne
- Migrated to the Superpower Institute and rebranded to **Open Electricity** — covering more than just the NEM
- Open source project from day one
- GitHub still at **github.com/opennem/opennem**

Open Electricity

Australian energy market data platform

- Open data for the **NEM** (eastern) + **WEM** (western) grids
- 5-minute generation, emissions, and price data since 1999
- **~1 billion records** and growing ~500k/day
- REST API + data exports for researchers, industry, and media



openelectricity.org.au

The Platform

openelectricity.org.au — interactive dashboards

api.openelectricity.org.au — public REST API

- Python / FastAPI backend
- PostgreSQL + ClickHouse data layer
- Real-time data ingestion from AEMO
- Open source: **github.com/opennem/opennem**

Australia's Electricity Grids

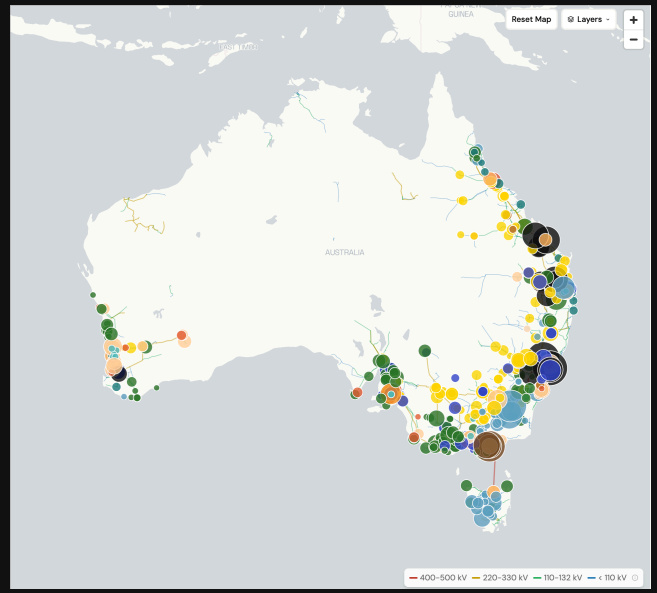
Both regulated by **AEMO** — 5-minute dispatch intervals

NEM — National Electricity Market

- Eastern states: NSW, QLD, VIC, SA, TAS
- Interconnected via transmission lines
- ~90% of Australia's electricity

WEM — Wholesale Electricity Market

- Western Australia only
- Completely isolated — no interconnectors



Grid Hierarchy

```
Network (NEM / WEM)
    └── Facility (power station)
            └── Unit (individual generator)
                    └── Fuel Technology
```

- NEM has ~**430** facilities, WEM has ~**60**
- Each facility has one or more **units** (generators)
- Each unit has a **fuel technology** type:
  - Coal, Gas, Wind, Solar, Hydro, Battery, ...
- Our data is recorded **per unit, per 5-minute interval**

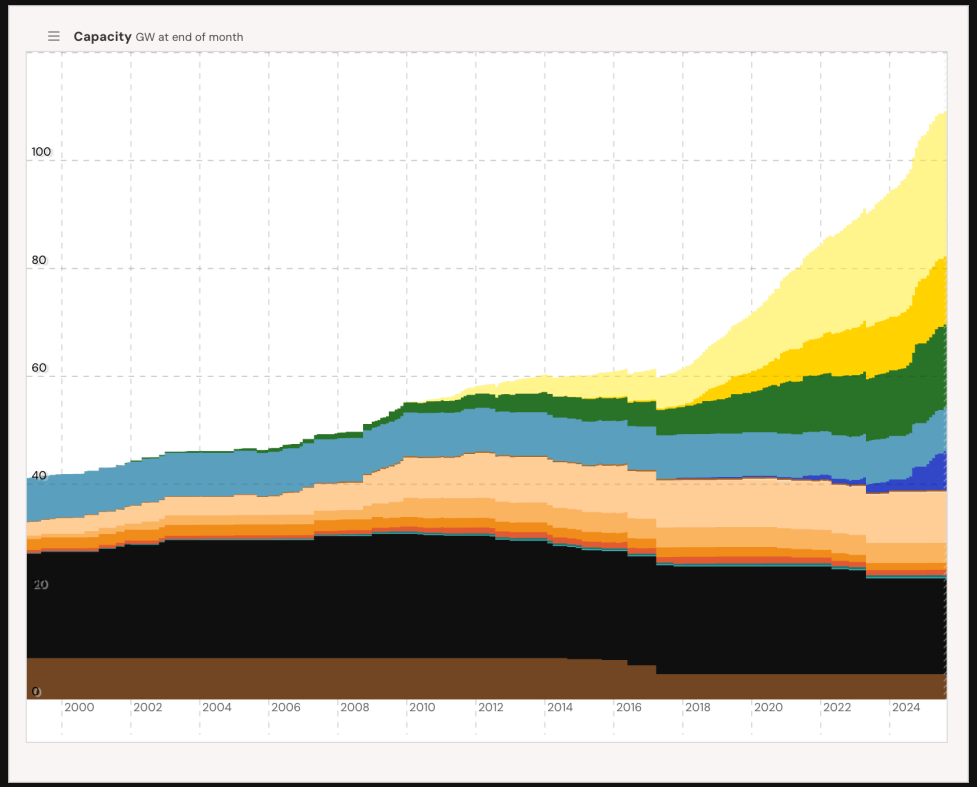
A Growing Grid

40 new facilities registered in 2025 alone

Type	Count
Battery	21
Solar	12
Wind	3
Gas / Bioenergy / Hydro	4

- More than **half** are battery storage
- Each new facility = more units = more 5-minute rows
- The dataset grows faster every year

A Changing Grid



Both **capacity** and **generation mix** are changing — many facilities now distributed across the grid

The Data

All 5-minute interval data from AEMO

Market data — per network, per region

- Price, demand, total generation
- Forecasts of each
- Power interchange between regions (interconnectors)

Generation data — per facility unit

- Generation output per 5-minute interval

Facility data — curated by Open Electricity

- Fuel technology type, capacity, location
- 100+ fields per facility

facility_scada — Generation Data

The bulk of the data — **~1B rows**

```
interval        DateTime        -- 5-minute interval
network_id      String          -- NEM / WEM
unit_code       String          -- generator DUID
generated       Float           -- MW output
energy          Float           -- MWh (calculated)
emissions       Float           -- tCO2e
market_value    Float           -- $ revenue
```

One row per unit, per 5-minute interval, since 1999

balancing_summary — Market Data

```
interval        DateTime        -- 5-minute interval
network_id      String          -- NEM / WEM
network_region  String          -- NSW1, QLD1, SA1, ...
price           Float           -- $/MWh spot price
demand          Float           -- MW total demand
generation      Float           -- MW total generation
price_forecast  Float           -- forecast price
demand_forecast Float           -- forecast demand
```

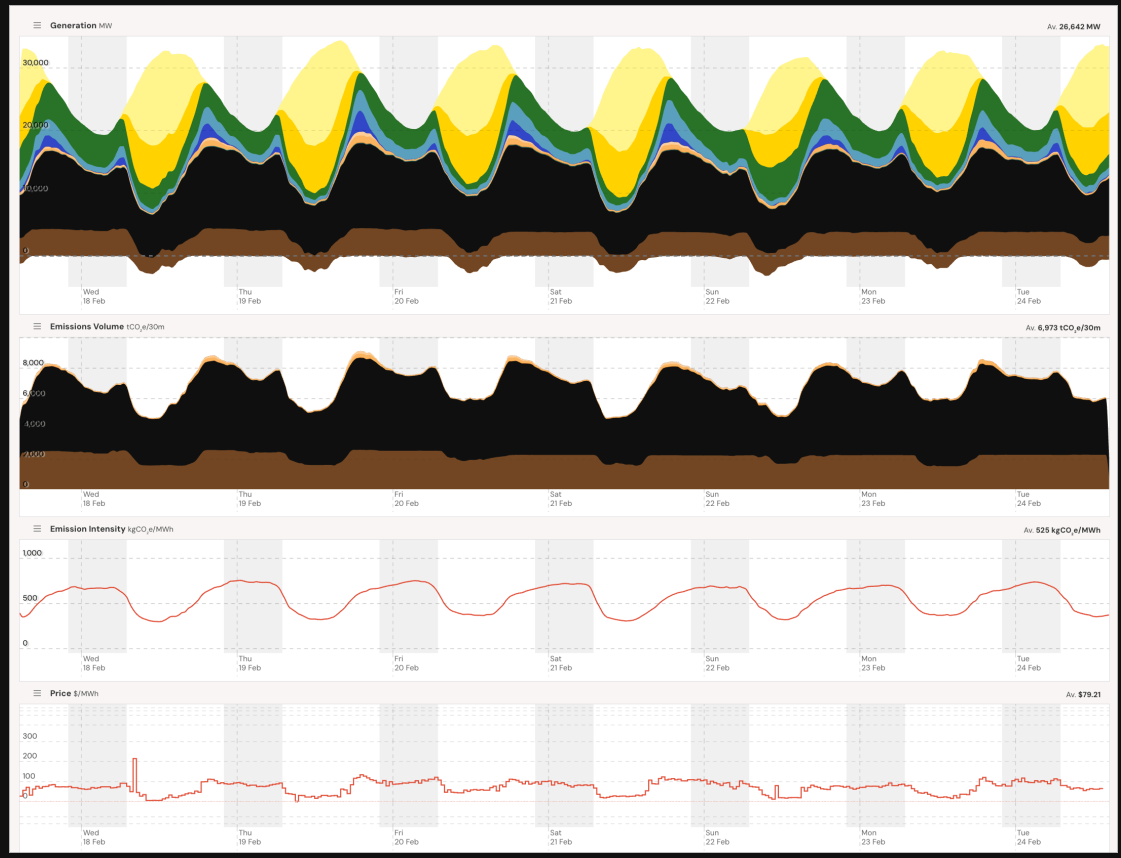
Per region, per 5-minute interval

facilities & units — Facility Data

```
facilities
    code            String      -- station code
    name            String      -- station name
    network_id      String      -- NEM / WEM
    network_region  String      -- NSW1, QLD1, ...

units
    code            String      -- DUID (generator ID)
    fueltech_id     String      -- coal, solar, battery, ...
    capacity        Float       -- registered capacity MW
    registered      DateTime    -- AEMO registration date
    location        Geography   -- lat/lng
```

~490 facilities, ~1200 units — curated by Open Electricity



Open Electricity

Default View — 7 Days by Fueltech

A single chart on openelectricity.org.au:

12

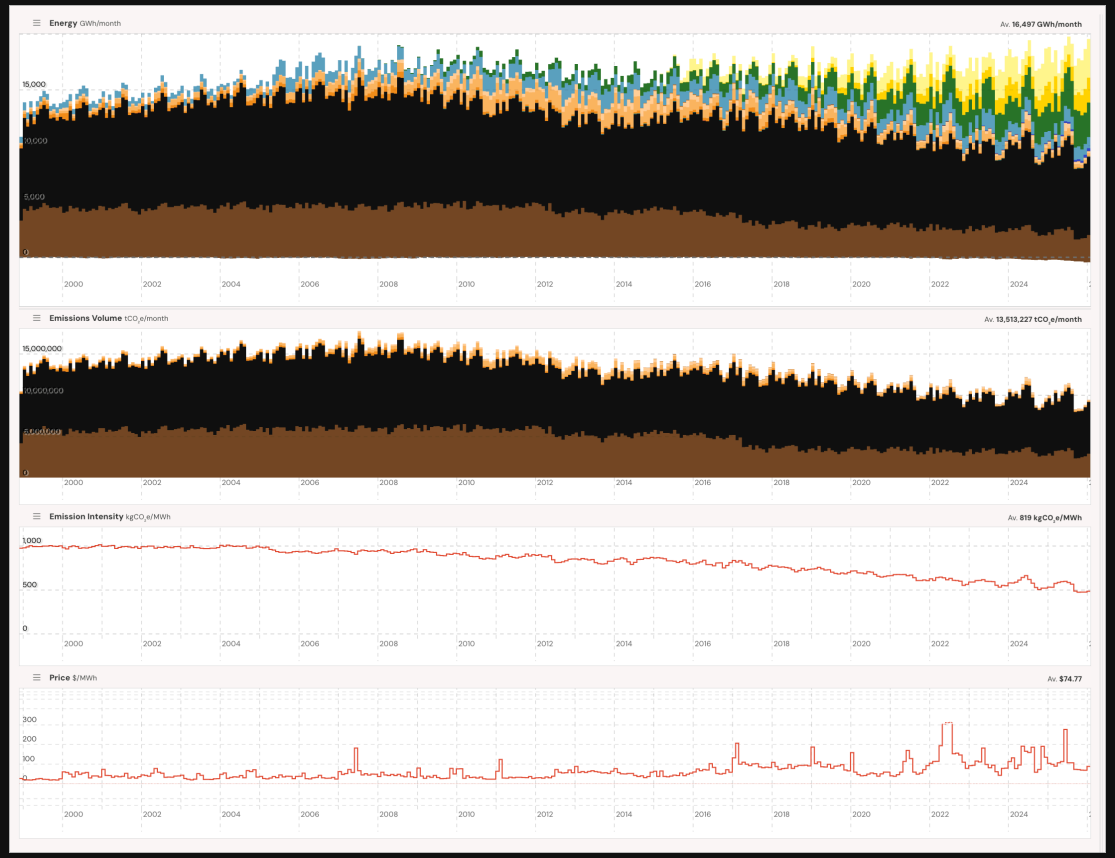
× 24 × 7 × 16

5-min intervals per hour

× hours per day
× days
× fuel technology groups

= 32,256 datapoints

Per region. Just for one chart.


Energy GWh/month
Av. 16,497 GWh/month
15,000
10,000
5,000
0
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
2024
Emissions Volume tCO2e/month
Av. 13,513,227 tCO2e/month
15,000,000
10,000,000
5,000,000
0
Emission Intensity kgCO2e/MWh
Av. 819 kgCO2e/MWh
1,000
500
0
Price $/MWh
Av. $74.77
300
200
100
0

All View — 1999 to Today

Every 5-minute interval for 27 years:

12

× 24 × 365 × 27 × 16

5-min intervals per hour

× hours per day
× days per year
× years (1999–2026)
× fuel technology groups

= ~45 million datapoints

Per region. Under 1 second.

The Challenge

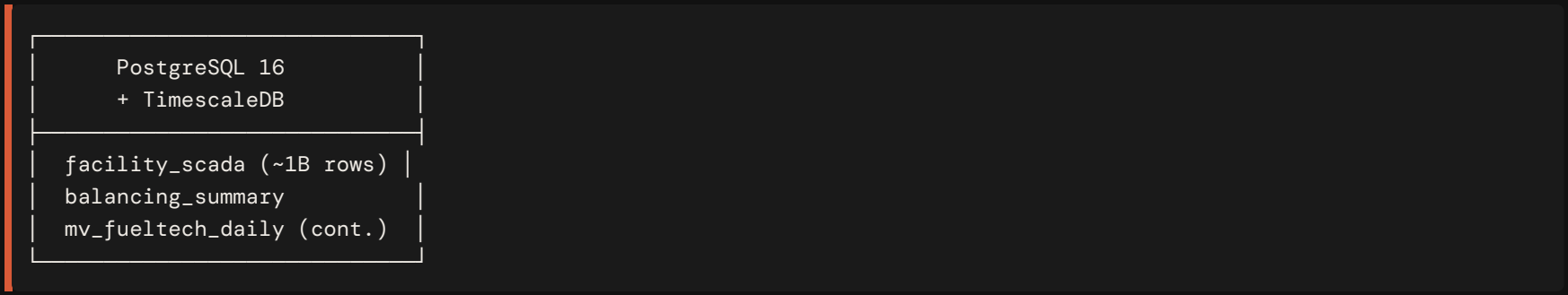
Metric	Value
Total records	~1B
Time span	1999 – present
Interval	5 minutes
Daily new rows	~500k

Pain: some queries can take 5 minutes+

Solution — Materialize!

- TimescaleDB continuous aggregates don't support `time_bucket_gapfill`
- No timezone handling in materialized views
- Queries on continuous aggregates are still **slow**
- Refreshing materialized views locks tables

Old Architecture



TimescaleDB gave us:

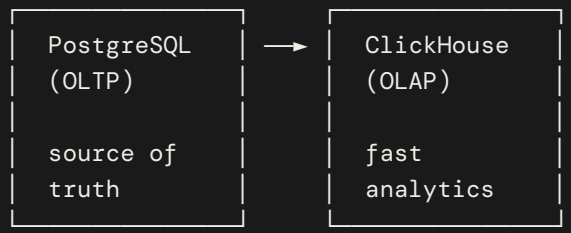
- `time_bucket_gapfill()`
- `interpolate()` / `locf()`
- Continuous aggregates

Actual Solution — Materialize with an OLAP

ClickHouse gives us:

- **Column store** — only reads the columns you query
- **Efficient compression** — ~1B rows in ~5GB on disk
- **Multiple engine types** — ReplacingMergeTree, SummingMergeTree, AggregatingMergeTree
- **Cheap materialized views** — pre-computed rollups that update automatically
- **Performance** — 45M datapoint queries in under 1 second

New Architecture



Best of both worlds.

ClickHouse Tables

unit_intervals — ~1B rows

- Per-unit 5-min generation / emissions / market value

market_summary

- Per-region 5-min price / demand

Both use `ReplacingMergeTree(version)`

MV Hierarchy

```
unit_intervals (5-min, ~1B)
    │
    ├── unit_intervals_daily_mv
    │
    ├── fueltech_intervals_mv
    │       └── fueltech_intervals_daily_mv
    │
    └── renewable_intervals_mv
            └── renewable_intervals_daily_mv
```

Query daily MV: **40ms** vs raw: **30s**

MergeTree Engine Types

Engine	What It Does	Use For
ReplacingMergeTree	Keeps highest version	Deduplication
SummingMergeTree	Sums numeric columns	sum(), count()
AggregatingMergeTree	Merges aggregate states	avg(), uniq()

Key insight: Replacing = deduplication, NOT aggregation

Benchmark — PostgreSQL Query

All-time energy by fueltech (1999–2026):

```
SELECT
    time_bucket('1 month', fs.interval) as month,
    u.fueltech_id,
    SUM(fs.energy) as total_energy
FROM facility_scada fs
JOIN units u ON fs.facility_code = u.code
WHERE fs.network_id IN ('NEM', 'AEMO_ROOFTOP')
    AND fs.is_forecast = false
    AND u.fueltech_id IS NOT NULL
GROUP BY 1, 2
ORDER BY 1, 2
```

Benchmark — ClickHouse Query

Same query in ClickHouse:

```
SELECT
    toStartOfMonth(interval) as month,
    fueltech_id,
    sum(energy) as total_energy
FROM unit_intervals FINAL
WHERE network_id IN ('NEM', 'AEMO_ROOFTOP')
    AND fueltech_id != ''
GROUP BY 1, 2
ORDER BY 1, 2
```

No JOIN needed — fueltech is denormalized into `unit_intervals`

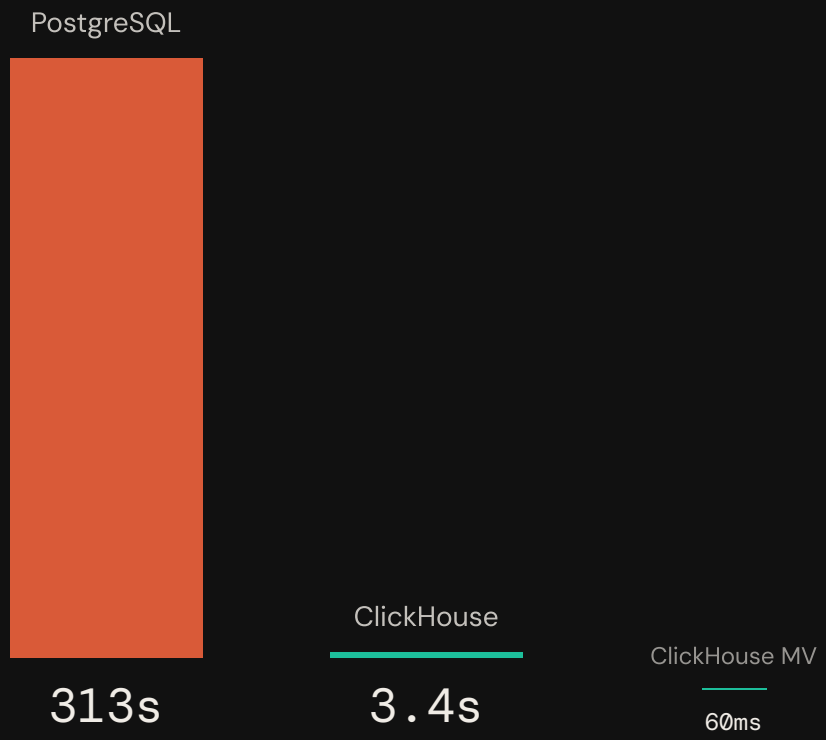
Benchmark — Results

Database	Time
PostgreSQL + TimescaleDB	**313s**
ClickHouse	**3.4s**

~92× faster

Same data. Same query. Similar server class.

Benchmark — Visualized



Server Specs

	PostgreSQL	ClickHouse
CPU	16 cores	4 cores
RAM	32 GB	4 GB
Disk	1.6 TB	200 GB

92× faster on a fraction of the hardware

Gotcha #1 — The FINAL Trap

```
-- BAD: Counts all versions
SELECT sum(energy) FROM unit_intervals;
-- Result: 200% of actual! (duplicates not merged)

-- GOOD: Only latest version
SELECT sum(energy) FROM unit_intervals FINAL;
-- Result: Correct value
```

Gotcha #2 — MV Auto-Population Trap

Auto-populating MVs + ReplacingMergeTree = **disaster**

1. New 5-min data → MV inserts partial daily aggregate
2. More data arrives → MV inserts another partial
3. ReplacingMergeTree keeps highest version
4. Highest version = NEWEST = **partial data!**

Result: Daily MV shows **40%** of actual energy

The Completeness-Based Version Hack

Problem: `max(version)` favors newest (partial) data

Solution: Version = completeness × 1B + version

```
toUInt64(count(distinct interval)) * 1000000000
    + max(version) as version
```

Source	Intervals	Version
Backfill	288	288,000,000,XXX
Auto-pop	10	10,000,000,XXX

More complete aggregate **always wins**.

Gotcha #3 — Timezones

```
-- BAD
interval DateTime64(3)
-- Returns: 2026-01-15 10:30:00.000
-- JS thinks: local time

-- GOOD
interval DateTime64(3, 'UTC')
-- Returns: 2026-01-15 10:30:00.000+00:00
-- JS knows: UTC
```

Zero storage cost. Just metadata.

Gotcha #4 — Memory

Can't load ~1B rows into memory.

Solution: Triple chunking

```
3-day time chunks
    ↓
20k record batches (server cursor)
    ↓
Polars DataFrame (fast transforms)
    ↓
ClickHouse batch insert
```

Force GC at 6GB threshold.

Gotcha #5 — ”Out of Memory”

Error: `Cannot reserve X MiB`

Not RAM — it’s **disk space**!

ClickHouse needs temp disk for:

- `OPTIMIZE TABLE FINAL`
- Large `GROUP BY`
- Merge operations

Need ~100GB free for backfills.

Gotcha #6 — Stale MVs

```
Day 1: Source 50% complete → MV shows 50%
Day 2: Source backfilled to 100%
Day 3: MV still shows 50%!
```

ReplacingMergeTree keeps old version.

Fix: Re-backfill + `OPTIMIZE TABLE FINAL`

Gotcha #7 — Timeouts

Operation	Timeout
User query	10s
Backfill batch	300s
OPTIMIZE FINAL	1000s

Match timeout to operation.

What Worked

1. **Feature flags** — toggle CH vs PG per endpoint
2. **Identical API output** — clients don't know which DB
3. **Each DB does its job** — PG gap-fills, CH aggregates
4. **MV hierarchy** — pre-computed rollups
5. **Completeness-based versioning** — backfills always win

What We'd Change

1. Document CH version requirements
2. Test MVs with partial data scenarios
3. Automated consistency checks (PG vs CH daily)
4. Use `FINAL` from day one
5. Understand MergeTree types **before** choosing

Migration Checklist

- `FINAL` on all ReplacingMergeTree queries
- `'UTC'` on all DateTime columns
- Chunk backfills (3-day + 20k batches)
- Monitor disk, not just RAM
- Re-backfill MVs after source fixes
- Document data source quirks
- Match timeouts to operations
- Feature flags for gradual rollout
- Completeness-based version for daily MVs

Key Takeaway

- PostgreSQL for **OLTP reliability**
- ClickHouse for **OLAP speed**
- Feature flags for **safe migration**
- Completeness-based version for **MVs**

No big-bang cutover needed.

Q&A

Open Electricity

Thank You

openelectricity.org.au

api.openelectricity.org.au

github.com/opennem/opennem

Nik Cubrilovic — Open Electricity