Just use Postgres

Victor Adossi

June, 2019

Roadmap

- What is Postgres?
- Why/Why not Postgres?
- Key/Value Data
- Document Storage
- Geospatial Data
- Message Queue
- Time Series Data

What is Postgres?

Postgres is the most advanced open source database that's ever existed. It's developed in the open, driven and maintained by the community.

There are a few large contributors in the space like **2nd Quadrant** and **Citus Data** (acquired by MS in January).

Some of the features that set postgres apart:

- Multi Version Concurrency Control (MVCC)
- Plugin system (indices, functionality)
- Process-per-connection model
- Elephant mascot

Reddit got to 1 billion users on a master-slave Postgres (scaling up rather than out, mostly)[1]

[1] http://highscalability.com/blog/2013/8/26/reddit-lessons-learned-from-mistakes-made-scaling-to-1-billi.html

Why/Why not Postgres?

Reliability - Postgres is rock solid

Performance - "fast enough" to *pretty darn fast*

Cloud vendor support - AWS RDS, Azure Database, GCP Cloud SQL

Open Source - You can see how it works

Why *not* Postgres?

No Vendor - No vendor to call[2]

Scaling Out - No official scale out story[3]

Learning Curve - Structured Query Language (SQL) can be difficult

Rigor - Transactional guarantees can eat into performance

[2] Lots of consultancies though (like 2nd Quadrant) which can help out

[3] PostgresXL does exist

Key/Value Data

Postgres makes a surprisingly good simple key value store. You're not going to beat Redis, but it's *probably* going to be fast enough!

```
CREATE UNLOGGED TABLE kv (
  id serial GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  key text NOT NULL,
  value jsonb,
  created_at timestamptz NOT NULL DEFAULT NOW()
);
```

Pluggable storage engines (the table access interface)[4] has landed, you could *actually* put Redis in your Postgres

[4] https://www.postgresql.org/docs/devel/tableam.html

Document Storage

```
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

CREATE TABLE docs (
  id uuid PRIMARY KEY DEFAULT uuid_generate_v4(),
  data jsonb,
  updated_at timestamptz NOT NULL DEFAULT NOW(),
  created_at timestamptz NOT NULL DEFAULT NOW()
);

-- GIN indexes massively speed up searches like:
-- SELECT * FROM docs WHERE data @> {"some_key": "some_value"}
CREATE INDEX docs_data_idx ON docs USING GIN (data);
```

Look into Postgres's full range of JSON operators[5]. SQL/JSON (JSONPath for SQL) is coming in 12[6].

[5] https://www.postgresql.org/docs/current/functions-json.html

[6] https://www.postgresql.org/docs/12/functions-json.html#FUNCTIONS-SQLJSON-PATH

Geospatial Data

Geographic Information System (GIS) data is the bread and butter of PostGIS[7]:

```
SELECT superhero.name
FROM city, superhero
WHERE ST_Contains(city.geom, superhero.geom)
AND city.name = 'Gotham';
```

Feature set and documentation for PostGIS is *extensive*.

[7] https://postgis.net

Message Queues

If all your application instances are connected to the database, why not have them communicate?

```
-- Create a channel named "virtual"
LISTEN virtual;

-- Notify with no payload
NOTIFY virtual;

-- notify with payload
NOTIFY virtual, 'This is the payload';
```

Maybe you don't need a NATS/RabbitMQ/NSQ/Kafka cluster *just* yet.

Want to go deeper? Try combining this feature with some `UNLOGGED` and `TEMPORARY` tables and build some data pipelines.

Time Series Data

You could build your own solution by using `PARTITIONs`, `UNLOGGED` tables, some `TRIGGERs`, but don't bother. Just use TimescaleDB[8].

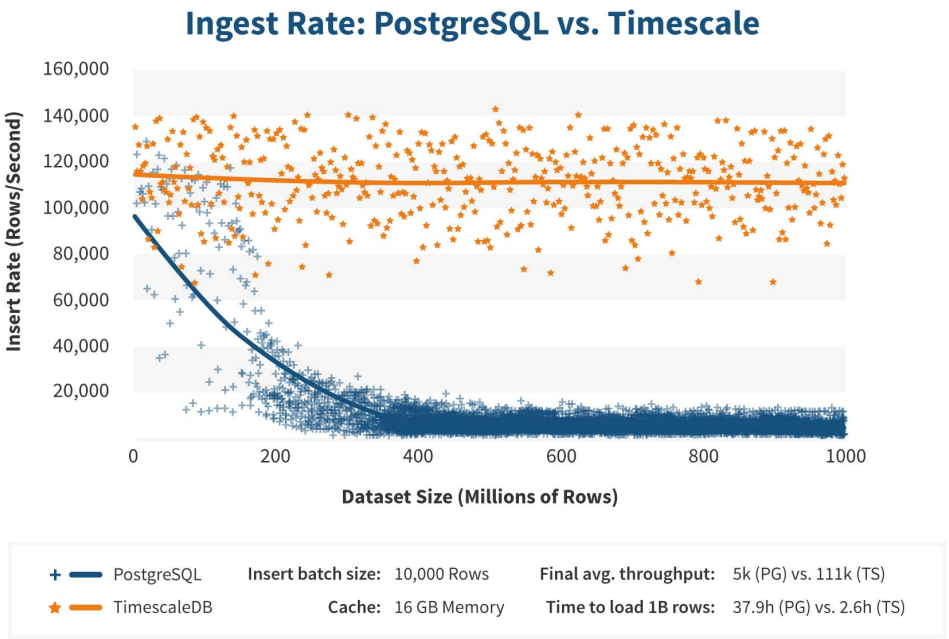
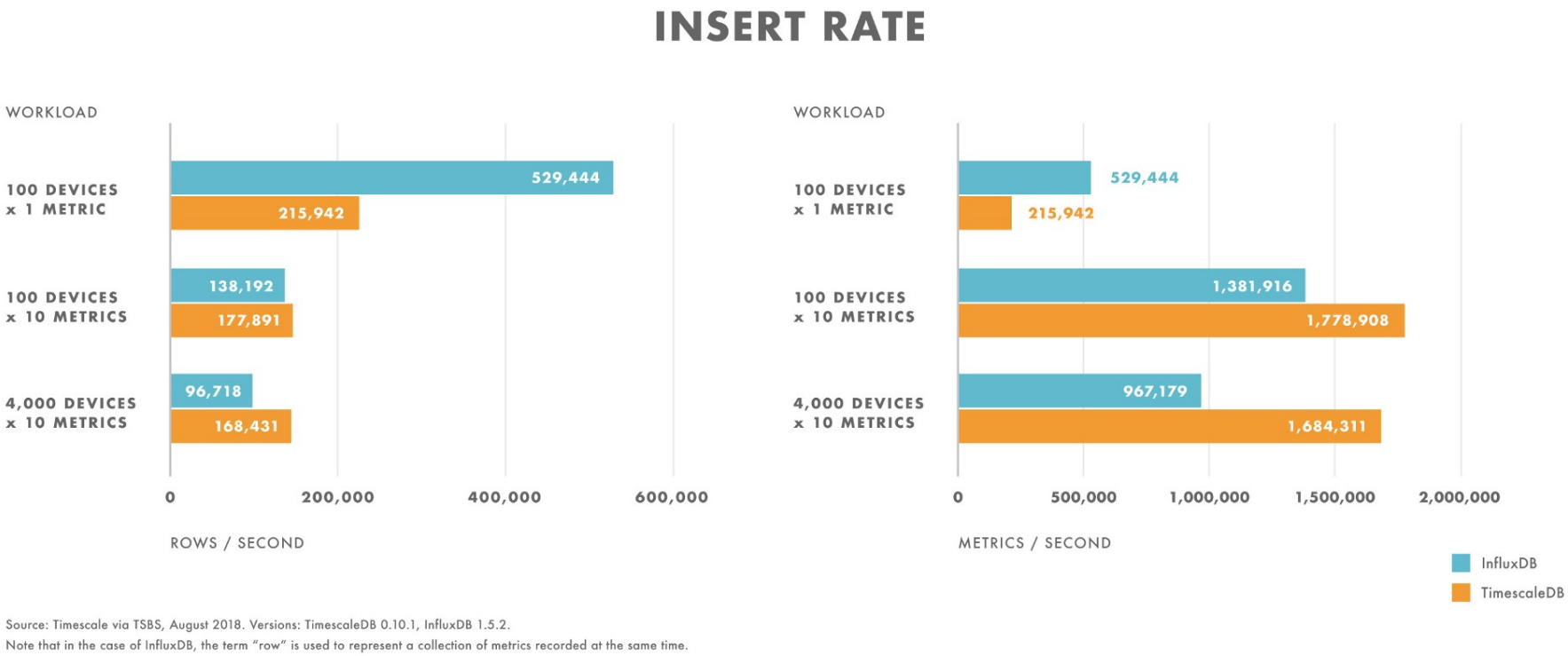


Figure 1: TimescaleDB insert performance on 1B inserts

[8] https://docs.timescale.com/v1.3/introduction

Time Series Data (continued)

TimescaleDB compares favorably to MongoDB[9] and InfluxDB[10].



Source: Timescale via TSBS, August 2018. Versions: TimescaleDB 0.10.1, InfluxDB 1.5.2.
Note that in the case of InfluxDB, the term "row" is used to represent a collection of metrics recorded at the same time.

[9] https://blog.timescale.com/how-to-store-time-series-data-mongodb-vs-timescaledb-postgresql-a73939734016

[10] https://blog.timescale.com/timescaledb-vs-influxdb-for-time-series-data-timescale-influx-sql-nosql-36489299877

So What?

Postgres may not be the best solution to your problem, but it's very often **good enough**.

Before introducing a new piece to your infrastructure, consider using your Postgres database to solve the problem.

The End

Thanks for listening

whoami

If you've got any corrections, complaints, or comments, feel free to reach me using the information below:

Victor Adossi (vados@vadosware.io, vados@gaisma.co.jp)

GPG: ED874DE957CFB552

I run a couple very small consultancies to support businesses in Japan and the USA:

- GAISMA G.K. (https://gaisma.co.jp)
- VADOSWARE LLC (https://vadosware.io)

Need help figuring out *how* you're going to use Postgres in your infrastructure? I can help with that.

Bloopers: Hot takes and tips

A bunch of things I think that are probably right:

- Use Gitlab
- Don't write ECMAscript (AKA Javascript) without Typescript
- Try Lisp & Haskell (separately?) at least once
- Try Rust more than once
- Never price by project*
- Don't build & deploy VMs on a greenfield project in 2019**

* Unless you've built the thing already and you are literally going to reskin it and the client has absolutely *no* new feature requests.

** Unless your VM in production is basically Container Linux