Navigating Postgres and Pgpool with a MySQL Expert's Perspective

Patrick Galbraith
Principal Engineer, Consultant

Patrick Galbraith

- 30+ years in tech
- Generalist
- Worked at US Navy (electronics), Slashdot, MySQL, HP Cloud, Oracle Cloud, etc
- Databases, Programming, SRE/Devops, Architecture
- patg@patg.net http://patg.net

Introduction

- **Overview**:
  - Brief introduction to PostgreSQL.
  - Introduction to Pgpool-II.
  - Importance of automation in database environments.
- **Objective**:
  - Understand PostgreSQL and Pgpool-II.
  - Learn how to automate their deployment and management using Ansible.
  - Compare PostgreSQL and MySQL from a DBA’s and developer’s viewpoint.

Architecture

- PostgreSQL: Emphasizes robustness, standards compliance.
  - More adherence originally other classic RDBMS
  - Gained more adoption in applications (cloud changed things)
- MySQL: Simplicity, wide adoption in web applications.
  - Started with inclusion with PHP w/MySQL client
  - Now much more in line with standards, CTE

Process Model

- MySQL threaded
- PostgreSQL: process
  - Work going on to make threaded
  - Yuri's work https://omnigres.com/

Data Integrity

- PostgreSQL
  - Strong focus on ACID properties
  - complex transactions
  - stored procedures
- MySQL
  - Offers different storage engines (InnoDB, Maria, CSV, …)
  - Although ACID compliant with similar to PostgreSQL

Replication

- PostgreSQL
  - Streaming WAL
    - Exact blocks
  - Logical WAL
    - SQL DDL/DML
  - Primary, hot standby replicas
- MySQL
  - Asynchronous binary or logical
  - Galera
  - Group

Backups

- PostgreSQL
  - Base-backup
  - Pg_dump (also Continuous PIT using logs replay)
  - Pgbackrest, barman
- MySQL
  - Xtrabackup
  - MySQL dump
  - File system

Load Balancing

- PostgreSQL
  - PgPool
  - PgBouncer
  - Haproxy
- MySQL
  - ProxySQL
  - Mysql router

Failover

- PostgreSQL with PgPool
  - Failover (healthcheck detects)
  - Follow primary
  - Failback
  - Multiple pgpool nodes with watchdog
- MySQL
  - Galera automatic
  - Async using promotion

Monitoring

- Percona Monitoring Management https://www.percona.com/software/database-tools/percona-monitoring-and-management
- Pgmetrics
- Pgwatch
- Coorot https://coroot.com
    - Prometheus compatible Open Source PostgreSQL Agent

Other differences

Syntax Differences:

- PostgreSQL’s stricter SQL standards might require query modifications.

Handling of NULLs and Empty Strings:

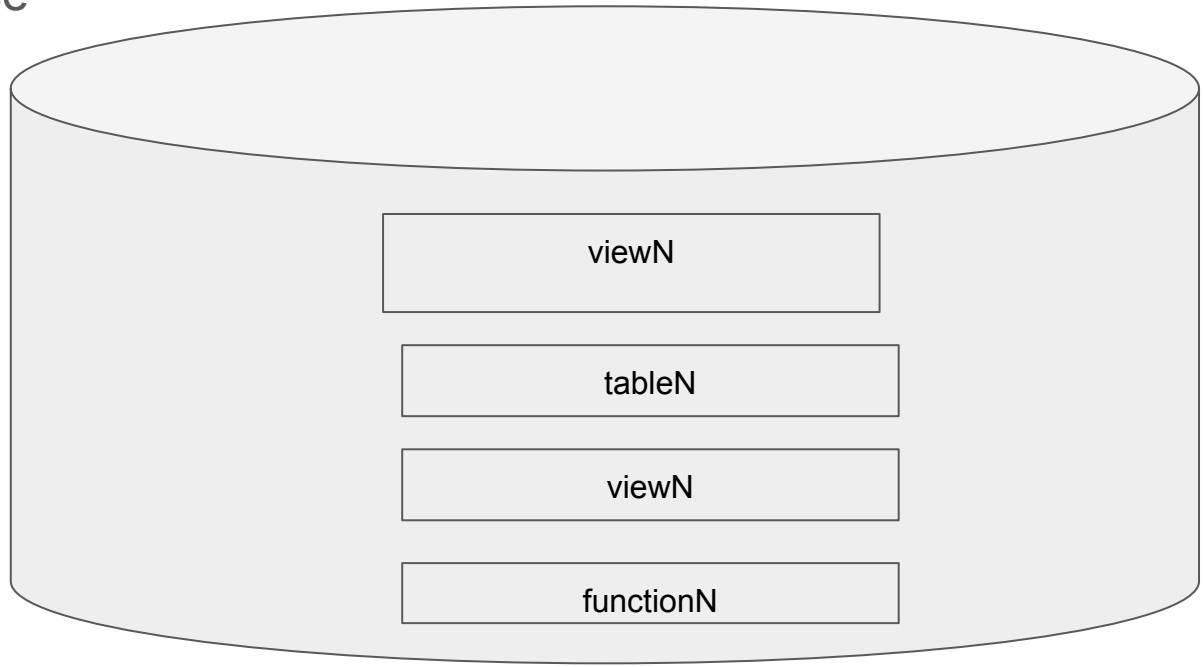
- PostgreSQL offers more advanced options like partial indexes and detailed NULL handling in Boolean contexts.

Indexing Strategies:

- PostgreSQL has more advanced indexing options (e.g., BRIN, GIN).

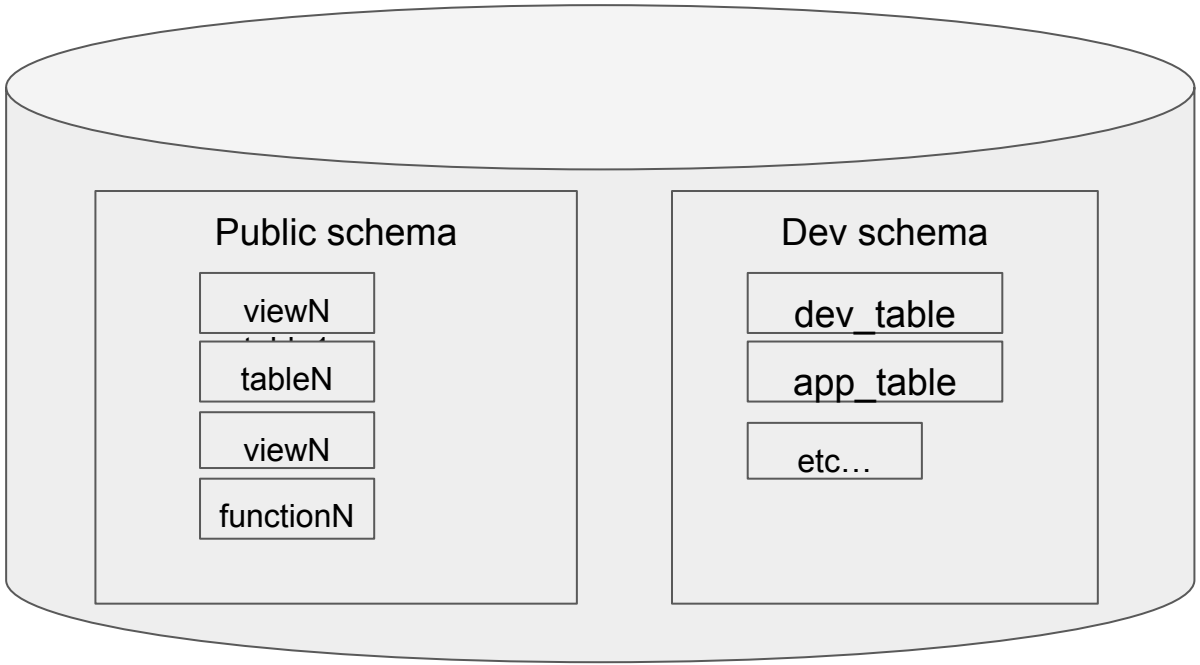
MySQL Object concepts

Database



PostgreSQL Object concepts

Database



Loadbalancing

PostgreSQL

- Pgpool II
  - R/W routing/load-balancing
  - HA
  - Connection pooling
  - Query Caching
- Pgbouncer
  - Session, Statement, Transaction Pooling

MySQL

- ProxySQL - caching, connection r/w routing, advanced rules, manageable via MySQL protocol w/backend sqlite
- MySQL Router - caching, simple connection routing, focus is MySQL ecosystem and InnoDB w/group replicaiton

Pgpool-II Overview

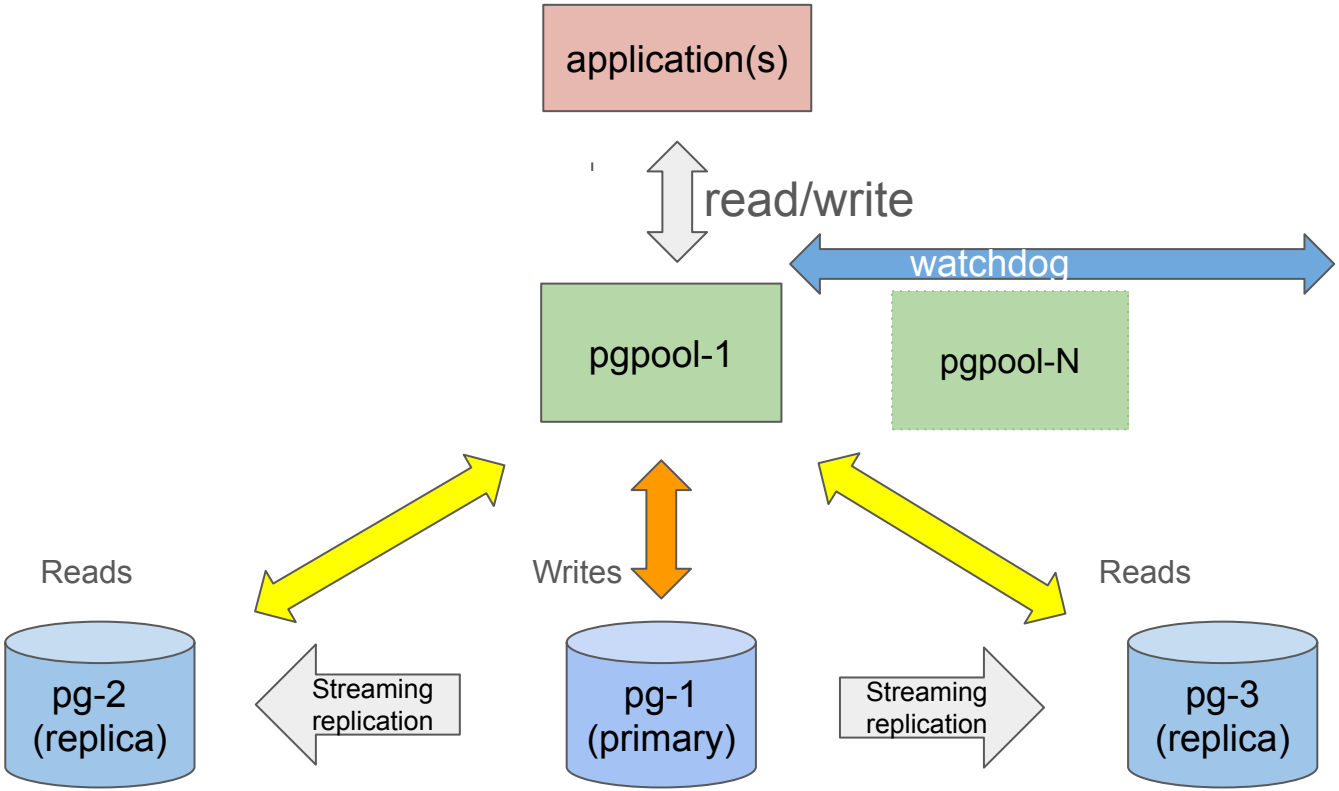
What is Pgpool-II?

- Middleware that sits between PostgreSQL clients and servers.
- Read/Write Query routing, including weighting
- Provides connection pooling, load balancing, replication, and high availability using healthcheck
- Query cache using Memcached
- Other tools: pgbouncer

Key Features:

- Connection pooling to reduce overhead.
- Load balancing for read queries.
- Automatic failover and failback.
- Online recovery of replicated nodes.

Basic PgPool setup



Automating PostgreSQL and Pgpool with Ansible

Why Automate?

- Consistency in deployments.
- Faster setup and scaling.
- Reduction in manual errors.
- Reduce toil

Ansible Overview:

- YAML-based configuration management and orchestration tool.
- Simple, agentless, and widely adopted in DevOps, using remote exec ssh

Playbooks:

- install_pg.yaml - Install postgres on 3 nodes
- Pg_slave_clone.yaml - Create base-backup and replica using the backup
- pg_users.yaml - creates non-system users (application users) specified in the inventory
- pgpass.yaml - writes out .pgpass file for postgres user on database nodes in inventory
- pg_hba.conf - writes out pg_hba and reloads postgresql
-

Ansible Playbooks for Pgpool-II:

- https://github.com/CaptTofu/pg-ansible-playbooks
- https://github.com/CaptTofu/terraform-pg-infra (on AWS test setup)

Ansible Modules for PostgreSQL:

- `postgresql_db`, `postgresql_user`, `postgresql_ext`, etc.

Other Open Source Options

- Various Kubernetes Operators
- Percona Everest
  https://docs.percona.com/everest/index.html

Summary

Recap

- Key differences between PostgreSQL and MySQL.
- Automate PostgreSQL and Pgpool-II using Ansible.
- Tips for transitioning from MySQL to PostgreSQL.

Final Thoughts

- You can build your own, though you will need to understand the thing you are
- Like Peter said earlier, building your own is harder than using DBaaS

I Can help!

- patg@patg.net
- https://patg.net

Pgpool usage

```
psql -p 9999 -h 127.0.0.1 -U app_user app
psql (15.7 - Percona Distribution, server 16.3 - Percona Distribution)
Type "help" for help.

app=> \l
                                    List of databases
   Name    |  Owner   | Encoding | Collate |  Ctype  | ICU Locale | Locale Provider |   Access privileges
-----------+----------+----------+---------+---------+------------+-----------------+-----------------------
 app       | app_user | UTF8     | C.UTF-8 | C.UTF-8 |            | libc            | =Tc/app_user         +
           |          |          |         |         |            |                 | app_user=CTc/app_user
 pgpool    | postgres | UTF8     | C.UTF-8 | C.UTF-8 |            | libc            | =Tc/postgres         +
           |          |          |         |         |            |                 | postgres=CTc/postgres+
           |          |          |         |         |            |                 | pgpool=CTc/postgres
 pmm       | pmm_user | UTF8     | C.UTF-8 | C.UTF-8 |            | libc            | =Tc/pmm_user         +
           |          |          |         |         |            |                 | pmm_user=CTc/pmm_user
 postgres  | postgres | UTF8     | C.UTF-8 | C.UTF-8 |            | libc            |
 template0 | postgres | UTF8     | C.UTF-8 | C.UTF-8 |            | libc            | =c/postgres          +
           |          |          |         |         |            |                 | postgres=CTc/postgres
 template1 | postgres | UTF8     | C.UTF-8 | C.UTF-8 |            | libc            | =c/postgres          +
           |          |          |         |         |            |                 | postgres=CTc/postgres
(6 rows)
app=> show pool_nodes;
 node_id | hostname | port | status | pg_status | lb_weight |  role   | pg_role | select_cnt | load_balance_node | replication_delay | replication_state | replication_sync_state |
last_status_change
---------+----------+------+--------+-----------+-----------+---------+---------+------------+-------------------+-------------------+-------------------+------------------------+---------------------
 0       | pg-1     | 5432 | up     | up        | 0.200000  | primary | unknown | 9          | false             | 0                 |                   |                        | 2024-08-23 18:37:41
 1       | pg-2     | 5432 | up     | up        | 0.400000  | standby | unknown | 0          | false             | 0                 |                   |                        | 2024-08-23 18:37:41
 2       | pg-3     | 5432 | up     | up        | 0.400000  | standby | unknown | 0          | true              | 0                 |                   |                        | 2024-08-23 18:37:41
(3 rows)
```

Pgpool load balancing distribution, writes

```
app=> show pool_backend_stats;
 node_id | hostname | port | status |  role   | select_cnt | insert_cnt | update_cnt | delete_cnt | ddl_cnt | other_cnt | panic_cnt | fatal_cnt |
error_cnt
---------+----------+------+--------+---------+------------+------------+------------+------------+---------+-----------+-----------+-----------+-----------
 0       | pg-1     | 5432 | up     | primary | 0          | 0          | 0          | 0          | 0       | 3         | 0         | 0         | 0
 1       | pg-2     | 5432 | up     | standby | 0          | 0          | 0          | 0          | 0       | 2         | 0         | 0         | 0
 2       | pg-3     | 5432 | up     | standby | 0          | 0          | 0          | 0          | 0       | 1         | 0         | 0         | 0
(3 rows)

postgres@pgpool-1:~$ (for x in $(seq 1 99); do psql -p 9999 -h 127.0.0.1 -U app_user app -c "insert into t1 values ($x)" -X
At; done)

node_id | hostname | port | status |  role   | select_cnt | insert_cnt | update_cnt | delete_cnt | ddl_cnt | other_cnt | panic_cnt | fatal_cnt |
error_cnt
---------+----------+------+--------+---------+------------+------------+------------+------------+---------+-----------+-----------+-----------+-----------
 0       | pg-1     | 5432 | up     | primary | 0          | 100        | 0          | 0          | 0       | 74        | 0         | 0         | 0
 1       | pg-2     | 5432 | up     | standby | 0          | 0          | 0          | 0          | 0       | 29        | 0         | 0         | 0
 2       | pg-3     | 5432 | up     | standby | 0          | 0          | 0          | 0          | 0       | 30        | 0         | 0         | 0
(3 rows)
```

Pgpool read balancing distribution

```
(for x in $(seq 1 10); do psql -p 9999 -h 127.0.0.1 -U app_user app -c "select * from t1 where id = $x"; done)


node_id | hostname | port | status |  role   | select_cnt | insert_cnt | update_cnt | delete_cnt | ddl_cnt | other_cnt | panic_cnt | fatal_cnt | error_cnt
---------+----------+------+--------+---------+------------+------------+------------+------------+---------+-----------+-----------+-----------+-----------
 0       | pg-1     | 5432 | up     | primary | 3          | 100        | 0          | 0          | 0       | 86        | 0         | 0         | 0
 1       | pg-2     | 5432 | up     | standby | 4          | 0          | 0          | 0          | 0       | 33        | 0         | 0         | 0
 2       | pg-3     | 5432 | up     | standby | 3          | 0          | 0          | 0          | 0       | 35        | 0         | 0         | 0
(3 rows)
```

PCP (admin) commands

```
$ pcp_node_info  -h 127.0.0.1
pg-1 5432 2 0.200000 up up primary unknown 0 none none 2024-08-27 10:16:35
pg-2 5432 2 0.400000 up up standby unknown 0 none none 2024-08-27 10:16:35
pg-3 5432 2 0.400000 up up standby unknown 0 none none 2024-08-27 10:16:35

$ pcp_health_check_stats  -h 127.0.0.1 -n 0
0 pg-1 5432 up primary 2024-08-27 10:16:35 15606 15606 0 0 0 0.000000 0 72 11 16.560682
2024-08-27 23:20:31 2024-08-27 23:20:31
$ pcp_<tab>
pcp_attach_node         pcp_node_count          pcp_proc_count          pcp_recovery_node
pcp_watchdog_info
pcp_detach_node         pcp_node_info           pcp_proc_info           pcp_reload_config
pcp_health_check_stats  pcp_pool_status         pcp_promote_node        pcp_stop_pgpool
```