Postgres Connection Pooling

Sunil Narain, Technical Director
Customer Success

29 April, 2021

EDB

Agenda page

- Pooling Introduction
- Why use pooling
- pgBouncer
- Pgpool-II
- Architecture Examples
- Conclusions
- Demo
- Q&A

What is Connection Pooling?

- Connection pooling is a method of :
  - Maintaining a pool of database connections in open state
  - Reusing an unused connection from the pool when a new request comes
  - Returning the connection back to the pool when application doesn’t need it
- Advantages:
  - limits number of connections to backend database
  - reduces the overhead of opening and closing connections and improves throughput

Why use pooling

- postgres architecture
  - Each connection to the database requires forking the database process
    - Individual process per client prevents a poorly behaving client from crashing the entire DB.
    - Forking has become more efficient in linux
- Modern apps tend to open/close connections rather than reusing them
  - Dev philosophy – “Open as late as possible, close as soon as possible”.
- Forking a process becomes expensive when transactions are very short
- Postgres performs well when number of concurrent connections are between 300-400.
- Pooling could be a benefit even with only a small number of connections expected

Connection Poolers

- Connection pooling provided by language libraries
- Middleware connection poolers
  - pgBouncer
  - pgPool

EDB

pgbouncer

pgBouncer

General Considerations

- Single threaded
- Multiple instances can be run on the same server
- Simple to configure
- Potentially a single point of failure

pgBouncer

Features

- Three pooling modes: Session, transaction, and statement
- Lightweight
- Easy to setup
- Passthrough Authentication
- Performance

pgBouncer

Configuration

- Plan to run multiple instances for larger client numbers.
- Plan for a maximum of 20-30 connections per instance
- Ensure OS is configured properly to support connections
- Statement mode needs to considered carefully before use
- Session and Transaction mode should be used accordingly

pgPool-II

Pgpool-II

General Considerations

- More than just a pooler
  - Failover, Heartbeat, query load balancer, replication
  - Additional functionality creates addition configuration complexity
- Supports HA architecture and operations
- Supports read scalability
  - The routing of SELECT statements could be better executed by the application as the routing of SELECT statements by Pgpool-II might prove unreliable.
- Can be useful as a load balancer

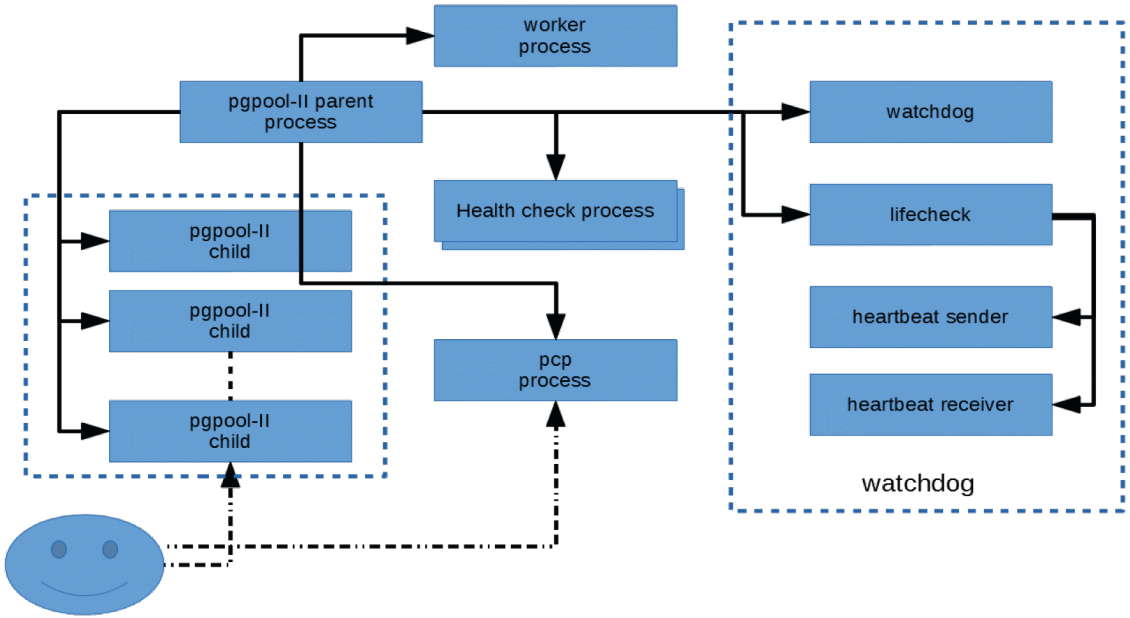
Pgpool-II

Running modes

- Streaming replication mode
- Native replication mode
- Logical replication mode
- Slony mode
- Snapshot isolation mode
- Raw mode

Pgpool-II

Architecture



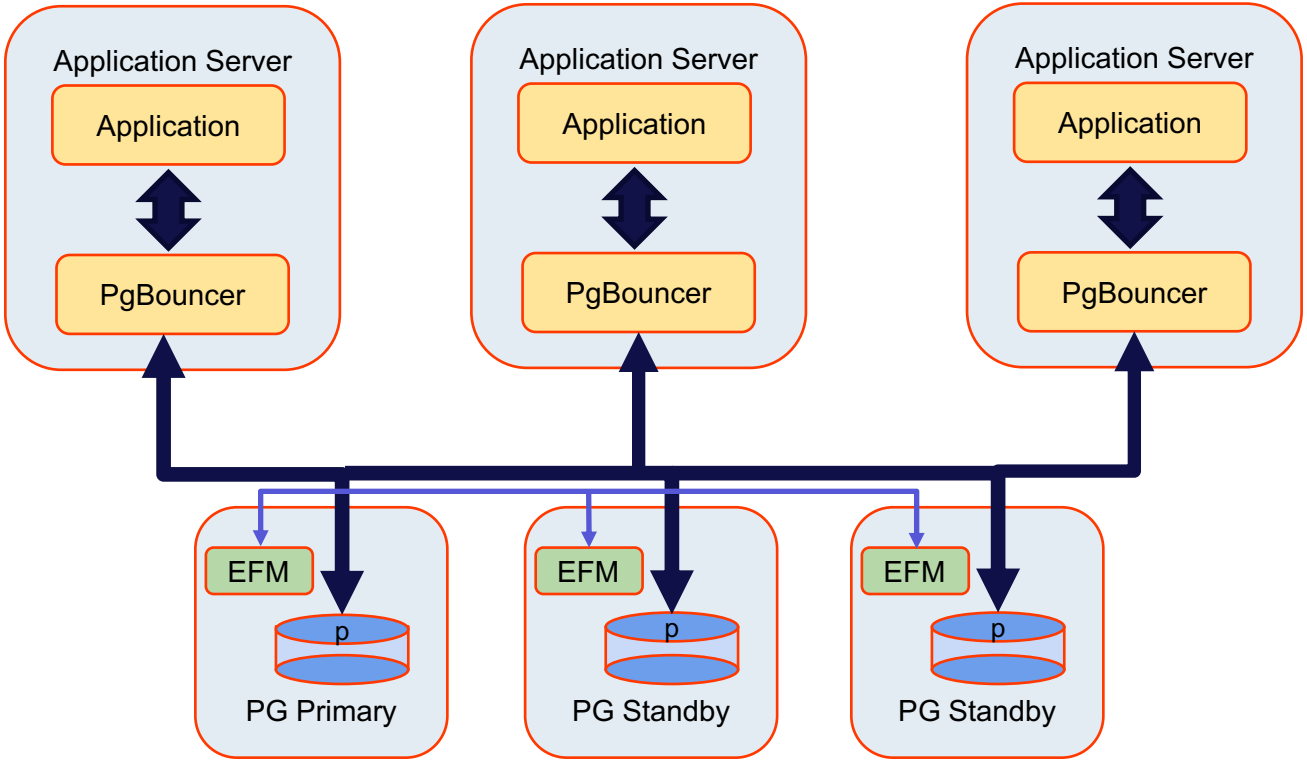
Pgpool-II

Configuration

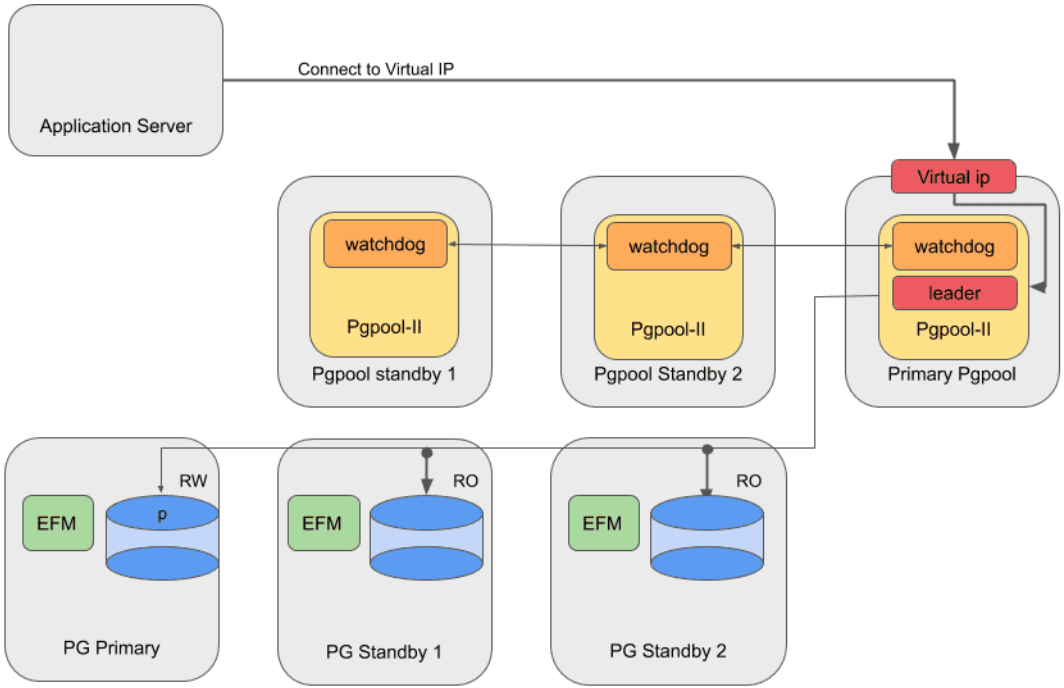
- Observe max_connections limit
- Ensure server and OS configuration supports number of connections
- Must understand the nature of the load the app generates

HA Architecture

HA with pgBouncer



HA with Pgpool-II



- https://www.enterprisedb.com/docs/efm/latest/efm_pgpool_ha_guide/03_components_ha_pgpool/
- https://www.enterprisedb.com/docs/efm/latest/efm_pgpool_ha_guide/

Conclusions

Pooling with Pgpool-II and pgBouncer

- pgBouncer and Pgpool-II can tolerate network outage, database restarts and even automated failover or controlled switchover.
- pgBouncer and Pgpool-II both provide connection pooling for Postgres databases on:
    - both Community Edition and EDB Postgres Advanced Server,
    - both Linux and Windows, and
    - can be built from source or downloaded from a repository.
- Using EDB Failover Manager with VIP capability, connections can be persistent from manual switchover from Master to Standby (e.g. during maintenance) and during automated failover from a Master to Standby using EDB EFM.

Summary

- Neither pgBouncer nor pgPool qualify as a magic bullet
- pgBouncer works well “out of the box” but will require heavy tuning for high-transaction systems
- pgPool has many [complex] settings, but also many features and with sufficient testing may be leveraged
- Direct access to the database, or allowing access via middleware such as jBoss, may be suitable and even desirable (e.g. where a steady number of long-lived active connections is maintained)
- Note that quiet failover is available with pgBouncer - see blogs for details
- Pgpool also has “watchdog”: VIP handling, failover and avoids “split brain”
- Setup of Pgpool is complex and has many potential points of failure!

Resources

- https://www.enterprisedb.com/blog/pgbouncer-tutorial-installing-configuring-and-testing-persistent-postgresql-connection-pooling
- https://www.enterprisedb.com/blog/pgbouncer-connection-pooling-what-do-when-persistent-connectivity-lost
- https://www.enterprisedb.com/blog/can-pgbouncer-handle-failover-new-machine
- https://www.enterprisedb.com/blog/can-pgbouncer-session-survive-everything-we-throw-it
- https://www.enterprisedb.com/blog/postgresql-pgpool-connection-pool-database-load
- https://www.enterprisedb.com/docs/pgpool/latest/
- https://www.pgpool.net/docs/latest/en/html/example-cluster.html
- https://www.enterprisedb.com/postgres-tutorials/why-you-should-use-connection-pooling-when-setting-maxconnections-postgres
- https://www.pgpool.net/docs/42/en/html/tutorial-arch.html#:~:text=Pgpool%2DII%20is%20a%20proxy,%22frontend%20and%20backend%20protocol%22.&text=It%20is%20responsible%20for%20forking,which%20accepts%20connections%20from%20clients.