crunchy data

Crunchy Bridge

Postgres at Scale: Running Multiple PgBouncers

Elizabeth Christensen
Nov 14, 2022 · 5 min read

Multi-PgBouncer is our affectionate in-house name for a special use case of running multiple instances of PgBouncer. PgBouncer is the venerable go-to tool for managing connection pooling at the database layer for Postgres. For some of our customers with notably large databases and high throughput, running multi-PgBouncer has been a great way to keep up with load and connections.

David Christensen recently did a talk at PGConf.NYC entitled "How to Tame a Mastodon: Lessons for PostgreSQL at Scale". Multi-bouncer was included as part of that talk and this topic got quite a bit of attention in the Q&A section.

If you're just getting familiar with connection pooling, check out Craig's excellent overview on "Your Guide to Connection Management in Postgres" or Karen Jex's video "Is there anything PgBouncer can't do".

In general, the benefits of using connection pooling are:

- Reduce the time it takes to establish a connection from an app server. Connection poolers keep Postgres connections open, so they can reuse or share an existing connection between multiple application connections. Particularly when using TLS or more expensive authentication methods, this can greatly increase app throughput.
- Reduce overall memory used. Every connection to the database takes up memory to create, maintain, and end the connection. When using a connection pooler, this takes up less overhead due to being able to have lighter-weight proxied connections inside of PgBouncer rather than full database connections to PostgreSQL.
- Reduce idle connections on the database. A common pattern before PgBouncer is many application connections that will make occasional queries, however your database works most efficiently when you have fewer connections being busier than many connections doing less work. PgBouncer allows you to adapt your database connections to your workload when using transaction mode.

A few notes about using pgBouncer at scale:

- PgBouncer is a single-threaded process which means it only uses a single CPU. Even if you have a larger server — say a 32 or 64 core server — you will never be able to devote more than 1 of your processors to PgBouncer.
- In general, a single PgBouncer can process up to 10,000 connections. 1,000 or

the application.

- Adjusting connection counts may also require you to adjust some system limits to allow PgBouncer to utilize the number of sockets required to support your desired number of application and database connections.
- This stuff isn't magic - if you are under-resourced in other ways, don't expect a database pooler to solve other performance issues. It's just one facet of a production database system.

Signs that you need more than one PgBouncer

Standard PgBouncer is packaged with Crunchy products and in general is sufficient for most use cases, including very large databases. However, when you run into some of the limits of a single PgBouncer instance, support for multi-PgBouncer can be a lifesaver.

You can tell that a single PgBouncer instance is having trouble keeping up when:

- PgBouncer's CPU usage is 100%.
- Application queries through PgBouncer wait times increase while Postgres itself is not similarly loaded.
- There is a mismatch in what the PgBouncer monitoring database shows in terms of active connections and what Postgres shows for the state for connections from PgBouncer.

When this happens, this is likely related to PgBouncer not being able to keep up with:

- the size of the result sets being returned from the database.

When we have seen this situation on Crunchy Bridge, profiling revealed that PgBouncer was spending the majority of its time copying data in/out of the SSL buffers from the upstream connections in preparation for sending back to the application connections.

If you want to do a quick check on how many connections you've got going:

```
# select state, count(*)
from pg_stat_activity
where backend_type = 'client backend'
group by 1;

        state        | count
---------------------+-------
 active              |   475
 idle                |     0
 idle in transaction |     0
(3 rows)
```

So here, if I've got `max_connections=500` , you're pretty close to using all of the available transactions, none are idle, this is a busy database, I might want multi-pgBouncer.

How Multi-PgBouncer works

Multi-PgBouncer works by running multiple PgBouncer instances via templated `systemd` service file, each listening on the same port, so incoming application traffic will spread between the running PgBouncer instances. Peter Eisentraut has an excellent writeup on how to set it up.

simple load balancing, since sharing this port is part of the service. This feature was added to pgBouncer in version 1.12, released in 2019.

This is part of our standard support option for Crunchy Bridge. Though not self-service it can be enabled by our support team. This service can also part of Crunchy Postgres for Kubernetes installations.

Coda

As far as downsides to multi-PgBouncer we haven't run across any big ones but some things to consider:

- Managing infrastructure you don't need is never a good idea
- Manage pool sizes and settings takes some extra care - our support can help with that
- PgBouncer is mostly commonly used individually so monitoring/metrics can be harder to reason about with multiple PgBouncers.

If you're thinking that multi-PgBouncer might be a solution for you, contact us, we'd love to chat.

Contributors: David Christensen, Daniel Farina, and Chris Bandy

Get more delivered to your inbox

Enter your email

Join The List

WRITTEN BY

Elizabeth Christensen

November 14, 2022 • More by this author

PRODUCTS

Crunchy Postgres

Crunchy Postgres for Kubernetes

Crunchy Bridge

Crunchy Certified PostgreSQL

Crunchy PostgreSQL for Cloud Foundry

Crunchy MLS PostgreSQL

Crunchy Spatial

SERVICES & SUPPORT

Enterprise PostgreSQL Support

Ansible Tower

Red Hat Partner

RESOURCES

Customer Portal

Software Documentation

Blog

Events

COMPANY

About Crunchy Data

Team

News

Careers

Contact Us

Newsletter

CRUNCHY DATA NEWSLETTER

Subscribe to the Crunchy Data Newsletter to receive Postgres content every month.

Enter your email

Join The List