Install Pgpool-II or PgBouncer before PostgreSQL for openQA instances, e.g. to be used on OSD

2019-08-08 14:48 - kraih

Status: New
Priority: Low
Assignee:
Category: Feature requests
Target version: future
Difficulty:

% Done: 0%
Estimated time: 0.00 hour

Description

We've recently seen quite a few log messages like this.

"Aug 07 02:19:14 openqa openqa[10258]: DBI connect('dbname=openqa','geekotest',...) failed: FATAL: remaining connection slots are reserved for non-replication superuser connections at /usr/lib/perl5/vendor_perl/5.18.2/Mojo/Pg.pm line 83."

And there are no obvious connection leaks in the webui. So it appears that at peak times we may be reaching the default PostgreSQL connection limit with gru jobs (100 minus a few reserved connections). But even when we are not at peak i count at least 68 active connections on OSD. Most are idle, and that is a lot of wasted resources.

PostgreSQL operates like an old Apache prefork server, so every idle connection requires a forked process. Pgpool-II (and the more minimalistic PgBouncer) are little daemons that run like a reverse proxy in front of PostgreSQL and that require almost no resources keeping even thousands of idle connections alive. Real PostgreSQL connections would only be used when there are active queries, and they would be shared.

Related issues:

Related to openQA Project - action #69355: [spike] redundant/load-balancing w... Resolved 2020-07-25

History

#1 - 2019-08-09 07:32 - okurz
so you are suggesting this to be installed but so far there is no reference to it within the packages or so, shouldn't we try to handle that upstream to e.g. for example "recommend" this?

Also, I have seen the message you gave as well. However shouldn't we rather investigate why this suddenly recently appeared and not "jump to conclusions"?

#2 - 2019-09-17 15:31 - okurz
On 2019-09-17 we had a similar incidence – see discussion in https://chat.suse.de/group/openqa-dev – which might be preventable by the suggestion mentioned in this ticket.

#3 - 2020-07-29 07:12 - okurz
- Priority changed from Normal to Low

#4 - 2020-07-30 09:46 - okurz
- Related to action #69355: [spike] redundant/load-balancing webui deployments of openQA added

#5 - 2020-10-22 13:36 - okurz
- Target version set to Ready

#6 - 2020-10-30 05:39 - okurz
- Target version changed from Ready to future

#7 - 2020-11-21 18:50 - okurz
- Project changed from openQA Infrastructure to openQA Project
Today I've tried both pgbouncer and pgpool-II in production with the legaldb. Here are some notes:

- **pgbouncer**: Not in Factory yet. Easy to install from package in `server:database:postgresql`. Simple configuration file with 3 basic modes of operation. Transaction mode pretty much just works for CRUD queries. Each transaction gets forwarded round robin to a backend connection in the pool as they become available. This would work great for the webui itself. Only problem is that pgbouncer does not support listen/notify, which the Minion job queue uses as an optimization to dequeue jobs faster. So it cannot be used for Gru. It has only minimal log output, but should be enough to track down issues.

- **pgpool-II**: Also not in Factory yet. Much harder to install, because the package doesn't appear to build currently for SLE/Leap 15.4, had to use the binary package from `server:database:postgresql`. Hard to configure, and the official documentation is close to useless. The operating modes here are very focused on replication with multiple replication strategies. Lots of options for external services to do failover and performance monitoring. Theoretically it supports listen/notify, but I ran into a problem where the job queue just froze, with queries not getting results anymore. Probably a misconfiguration on my part, but pgpool-II is clearly not a component you just add to your setup without serious preparation. The main focus seems to be failover for multiple replicated backend servers.

On the plus side, both packages from the devel project have .service files and example configs.

Update: After some more experimentation it seems pgpool-II works pretty well with the Minion job queue when these settings are used with one configured backend server: `backend_clustering_mode = 'raw', load_balance_mode = off, use_watchdog = off, memory_cache_enabled = off.`