Quick Links

- Community
- Contributors
- Mailing Lists
- IRC
- Local User Groups
- Events
- International Sites

Let's make PostgreSQL multi-threaded

From: Heikki Linnakangas <hlinnaka(at)iki(dot)fi>
To: pgsql-hackers <pgsql-hackers(at)postgresql(dot)org>
Subject: Let's make PostgreSQL multi-threaded
Date: 2023-06-05 14:51:57
Message-ID: 31cc6df9-53fe-3cd9-af5b-ac0d801163f4@iki.fi
Views: Raw Message | Whole Thread | Download mbox | Resend email
Thread: 2023-06-05 14:51:57 from Heikki Linnakangas <hlinnaka(at)iki(dot)fi>
Lists: pgsql-hackers

```
I spoke with some folks at PGCon about making PostgreSQL multi-threaded,
so that the whole server runs in a single process, with multiple
threads. It has been discussed many times in the past, last thread on
pgsql-hackers was back in 2017 when Konstantin made some experiments [0].

I feel that there is now pretty strong consensus that it would be a good
thing, more so than before. Lots of work to get there, and lots of
details to be hashed out, but no objections to the idea at a high level.

The purpose of this email is to make that silent consensus explicit. If
you have objections to switching from the current multi-process
architecture to a single-process, multi-threaded architecture, please
speak up.

If there are no major objections, I'm going to update the developer FAQ,
removing the excuses there for why we don't use threads [1]. And we can
start to talk about the path to get there. Below is a list of some
hurdles and proposed high-level solutions. This isn't an exhaustive
list, just some of the most obvious problems:

# Transition period

The transition surely cannot be done fully in one release. Even if we
could pull it off in core, extensions will need more time to adapt.
There will be a transition period of at least one release, probably
more, where you can choose multi-process or multi-thread model using a
GUC. Depending on how it goes, we can document it as experimental at first.

# Thread per connection

To get started, it's most straightforward to have one thread per
connection, just replacing backend process with a backend thread. In the
future, we might want to have a thread pool with some kind of a
scheduler to assign active queries to worker threads. Or multiple
threads per connection, or spawn additional helper threads for specific
tasks. But that's future work.

# Global variables

We have a lot of global and static variables:

$ objdump -t bin/postgres | grep -e "\.data" -e "\.bss" | grep -v
"data.rel.ro" | wc -l
1666

Some of them are pointers to shared memory structures and can stay as
they are. But many of them are per-connection state. The most
straightforward conversion for those is to turn them into thread-local
variables, like Konstantin did in [0].

It might be good to have some kind of a Session context struct that we
pass everywhere, or maybe have a single thread-local variable to hold
it. Many of the global variables would become fields in the Session. But
that's future work.

# Extensions

A lot of extensions also contain global variables or other things that
break in a multi-threaded environment. We need a way to label extensions
that support multi-threading. And in the future, also extensions that
*require* a multi-threaded server.

Let's add flags to the control file to mark if the extension is
thread-safe and/or process-safe. If you try to load an extension that's
not compatible with the server's mode, throw an error.

We might need new functions in addition _PG_init, called at connection
startup and shutdown. And background worker API probably needs some changes.

# Exposed PIDs

We expose backend process PIDs to users in a few places.
pg_stat_activity.pid and pg_terminate_backend(), for example. They need
to be replaced, or we can assign a fake PID to each connection when
running in multi-threaded mode.

# Signals

We use signals for communication between backends. SIGURG in latches,
and SIGUSR1 in procsignal, for example. Those primitives need to be
rewritten with some other signalling mechanism in multi-threaded mode.
In principle, it's possible to set per-thread signal handlers, and send
a signal to a particular thread (pthread_kill), but I think it's better
to just rewrite them.

We also document that you can send SIGINT, SIGTERM or SIGHUP to an
individual backend process. I think we need to deprecate that, and maybe
come up with some convenient replacement. E.g. send a message with
backend ID to a unix domain socket, and a new pg_kill executable to send
those messages.

# Restart on crash

If a backend process crashes, postmaster terminates all other backends
and restarts the system. That's hard (impossible?) to do safely if
everything runs in one process. We can continue have a separate
postmaster process that just monitors the main process and restarts it
on crash.

# Thread-safe libraries

Need to switch to thread-safe versions of library functions, e.g.
uselocale() instead of setlocale().

The Python interpreter has a Global Interpreter Lock. It's not possible
to create two completely independent Python interpreters in the same
process, there will be some lock contention on the GIL. Fortunately, the
python community just accepted https://peps.python.org/pep-0684/. That's
exactly what we need: it makes it possible for separate interpreters to
have their own GILs. It's not clear to me if that's in Python 3.12
already, or under development for some future version, but by the time
we make the switch in Postgres, there probably will be a solution in
cpython.

At a quick glance, I think perl and TCL are fine, you can have multiple
interpreters in one process. Need to check any other libraries we use.

[0]
https://www.postgresql.org/message-id/flat/9defcb14-a918-13fe-4b80-a0b02ff85527%40postgrespro.ru

[1]
https://wiki.postgresql.org/wiki/Developer_FAQ#Why_don.27t_you_use_raw_devices.2C_async-I.2FO.2C_.3Cinsert_your_favorite_wizz-bang_feature_here.3E.3F

--
Heikki Linnakangas
Neon (https://neon.tech)
```

Responses

- Re: Let's make PostgreSQL multi-threaded at 2023-06-05 15:18:27 from Tom Lane
- Re: Let's make PostgreSQL multi-threaded at 2023-06-05 15:28:50 from Tristan Partin
- Re: Let's make PostgreSQL multi-threaded at 2023-06-05 17:10:52 from Bruce Momjian
- Re: Let's make PostgreSQL multi-threaded at 2023-06-06 13:40:28 from Robert Haas
- Re: Let's make PostgreSQL multi-threaded at 2023-06-06 20:14:41 from Greg Stark
- Re: Let's make PostgreSQL multi-threaded at 2023-06-07 13:08:38 from Ashutosh Bapat
- Re: Let's make PostgreSQL multi-threaded at 2023-06-07 21:30:17 from Andres Freund
- Re: Let's make PostgreSQL multi-threaded at 2023-06-10 18:01:47 from Hannu Krosing
- Re: Let's make PostgreSQL multi-threaded at 2023-06-10 22:53:24 from James Addison

Browse pgsql-hackers by date

	From	Date	Subject
Next Message	David G. Johnston	2023-06-05 14:55:57	Re: QUAL Pushdown causes ERROR on syntactically and semantically correct SQL Query
Previous Message	Hans Buschmann	2023-06-05 14:40:35	QUAL Pushdown causes ERROR on syntactically and semantically correct SQL Query