Arvados - Bug #18488

[controller] does not release pg_advisory_lock($1) when it fails to start

11/29/2021 03:23 PM - Ward Vandewege

Status:	Resolved	**Start date:**	12/02/2021
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

This is bad because systemd restarts controller and after a number of tries, postgresql runs out of database connections.

Basically; it seems that postgresql does not kill waiting lock requests when the connection that the request came in on is severed abruptly.

This easy to replicate with two simultaneous psql connections (e.g. `select pg_advisory_lock(1);`). Remove the one that is waiting on the lock with `kill -9` (so it does not clean up the connection). Then check in the other one:

```
 SELECT pid, age(clock_timestamp(), query_start), state, usename, query
FROM pg_stat_activity
WHERE query != '<IDLE>' AND query NOT ILIKE '%pg_stat_activity%' AND state != 'idle'
ORDER BY query_start desc;
   pid   |       age       | state  | usename  |             query
---------+-----------------+--------+----------+------------------------------
 3924760 | 00:25:23.843543 | active | postgres | select pg_advisory_lock(1);
(1 row)
```

Presumably we could switch to the 'try' variant and retry occasionnaly to avoid this problem.

Subtasks:

Task # 18512: Review 18488-nonblocking-advisory-lock	**In Progress**

Related issues:

Related to Arvados - Bug #18487: controller fails to start when vocab propert...	**Resolved**	**01/20/2022**

Associated revisions

Revision 2bd768d4 - 12/03/2021 03:44 PM - Tom Clegg

Merge branch '18488-nonblocking-advisory-lock'

fixes #18488

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tom@curii.com>

History

#1 - 11/29/2021 03:24 PM - Ward Vandewege

- Related to Bug #18487: controller fails to start when vocab properties file is invalid (e.g. duplicate synonyms) added

#2 - 11/29/2021 03:24 PM - Ward Vandewege

- Description updated

#3 - 11/29/2021 10:29 PM - Ward Vandewege

- Description updated

#4 - 11/30/2021 04:06 PM - Tom Clegg

- Release set to 48

- Assigned To set to Tom Clegg

- Status changed from New to In Progress

18488-nonblocking-advisory-lock @ 9ed314b7a585970c03c87959286fc1e582d769f7 -- developer-run-tests: #2823 icon?job=developer-run-tests&amp;build=2823

#5 - 12/01/2021 08:09 AM - Carlos Fenoy

Can this be changed so that the TrashSweep only takes the lock before starting the sweep and frees it once finished?

I don't see a reason to hold a lock for the lifetime of the controller when it's only supposed to prevent multiple TrashSweep processes from running concurrently.

Holding a lock keeps a transaction opened which affects the behaviour of VACUUM in PostgreSQL and prevents it from freeing data deleted after the lock was acquired.

#6 - 12/01/2021 02:57 PM - Peter Amstutz

- Target version deleted (2021-12-08 sprint)

- Assigned To deleted (Tom Clegg)

Carlos Fenoy wrote:

> Can this be changed so that the TrashSweep only takes the lock before starting the sweep and frees it once finished?
>
> I don't see a reason to hold a lock for the lifetime of the controller when it's only supposed to prevent multiple TrashSweep processes from running concurrently.
>
> Holding a lock keeps a transaction opened which affects the behaviour of VACUUM in PostgreSQL and prevents it from freeing data deleted after the lock was acquired.

Is it holding the lock that keeps a transaction, or is it the other blocked lock attempts that are considered open transactions?

#7 - 12/01/2021 02:58 PM - Tom Clegg

The purpose of this lock isn't only to prevent concurrent trash sweeps, it's to ensure only one controller process does periodic trash sweeps. E.g., if trash sweep interval is 5 minutes and there are 5 controller processes/hosts, we want 1 trash sweep every 5 minutes, not 5.

We are planning to use a similar long-lived lock to ensure only one keep-balance process runs at a time, etc. If the PostgreSQL feature does not work as a long-lived lock, we'll need to find something else. But it seems a bit odd that the advisory lock feature would have such a significant side effect that (afaics) isn't mentioned in the docs.

Re keeping a transaction open, this does not use a transaction, and it isn't associated with any table, so I don't think it would prevent VACUUM. FWIW I tried it in pg console and "lock table" blocked a vacuum but pg_advisory_lock() did not.

#8 - 12/01/2021 03:39 PM - Carlos Fenoy

You can see from the output of the following queries that the vacuum is blocked by oldest xmin: 129621260 which corresponds to advisory_lock

It could be that the problem is the "pending" lock and not the acquired lock.

```
arvados_production=> select
 pid, age(clock_timestamp(), query_start), state, usename, backend_xid, backend_xmin, query from
 pg_stat_activity WHERE query != '<IDLE>' AND query NOT ILIKE '%pg_stat_activity%' AND state!='idle';
  pid  |           age           | state  | usename | backend_xid | backend_xmin |             query
-------+-------------------------+--------+---------+-------------+--------------+-----------------------------
 17088 | 1 day 16:26:51.612997 | active | arvados |             |    129621260 | SELECT pg_advisory_lock($1)
(1 row)

arvados_production=> vacuum verbose materialized_permissions ;
INFO:  vacuuming "public.materialized_permissions"
INFO:  launched 2 parallel vacuum workers for index cleanup (planned: 2)
INFO:  "materialized_permissions": found 0 removable, 1196424 nonremovable row versions in 15145 out of 35269
 pages
DETAIL:  1193 dead row versions cannot be removed yet, oldest xmin: 129621260
...
```

#9 - 12/01/2021 05:29 PM - Tom Clegg

Yes, it seems plausible that a *waiting* pg_advisory_lock() statement counts as an open transaction for vacuum purposes (after all, under the hood a single statement running outside a transaction is really a single-statement transaction) but vacuum is unaffected by the fact of an advisory lock being held in an open session.

In that case, the change in note-4 would resolve the issue.

Regardless, we discussed this relative to our longer term plans and decided we should use a different locking strategy anyway and we plan to have a new implementation in 2.3.2 that does not use pg_advisory_lock at all.

#10 - 12/02/2021 04:59 PM - Ward Vandewege

Tom Clegg wrote:

> 18488-nonblocking-advisory-lock @ 9ed314b7a585970c03c87959286fc1e582d769f7 -- developer-run-tests: #2823 icon?job=developer-run-tests&amp;build=2823

LGTM thanks.

#11 - 12/03/2021 03:55 PM - Tom Clegg

- % Done changed from 0 to 100

- Status changed from In Progress to Resolved

Applied in changeset arvados-private:commit:arvados|2bd768d4d0b06a2a1d3e3ce95ab686164b1d713a.