Reviewing PostgreSQL Patches for Fun and Profit

David Steele
Crunchy Data

PostgresOpen 2016
September 14, 2016

The Goal

How can we economize committer time?

The Purpose of Patch Review

The purpose of patch review is to answer the following questions:

- Does the patch work as specified?
- Is it useful and needed?
- Does the functionality already exist?
- Are documentation and tests included?
- Does it follow applicable SQL or community standards?
- Is performance acceptable?

How can we help?

- It is perfectly OK to only review parts of a patch, or only go through certain review steps.
- PostgreSQL is a community project and all contributors have different strengths. Some contributors might excel at documentation while others have a variety of platforms available for performance testing.
- No contribution is too small, especially to start.

The Commitfest App

- Located at `https://commitfest.postgresql.org`
- Shows past, current, and future commitfests.

Commitfests

The following commitfests exist in the system. New patches should be submitted to commitfest 2016-09.

- 2017-03 (Future - 2017-03-01 - 2017-03-31)
- 2017-01 (Future - 2017-01-01 - 2017-01-31)
- 2016-11 (Future - 2016-11-01 - 2016-11-30)
- 2016-09 (Open - 2016-09-01 - 2016-09-30)
- 2016-03 (Closed - 2016-03-01 - 2016-03-31)
- 2016-01 (Closed - 2016-01-02 - 2016-02-08)
- 2015-11 (Closed - 2015-11-01 - 2015-11-30)
- 2015-09 (Closed - 2015-09-01 - 2015-09-30)
- 2015-07 (Closed - 2015-07-01 - 2015-07-31)
- 2015-02 (Closed - 2015-02-15 - 2015-05-15)
- 2014-12 (Closed - 2014-12-15 - 2015-01-15)

The Commitfest App - Patch List

Commitfest 2016-09

Search/filter | Shortcuts ▾ | New patch

Status summary: Needs review: 76. Waiting on Author: 1. Ready for Committer: 8. Committed: 27. Rejected: 2. Total: 114.

Active patches

Patch ↓	Status	Author	Reviewers	Committer	Latest activity	Latest mail
Bug Fixes						
Fix the optimization to skip WAL-logging on table created in same transaction	Needs review	Heikki Linnakangas (heikki), Michael Paquier (michael-kun)	Michael Paquier (michael-kun)	heikki	2016-04-06 06:11	2016-04-06 06:11
OOM in libpq and infinite loop with getCopyStart()	Needs review	Michael Paquier (michael-kun)	Aleksander Alekseev (a.alekseev)		2016-04-06 06:07	2016-04-19 07:19
Flush slot confirmations on checkpoint	Needs review	Craig Ringer (ringerc)	Stas Kelvich (kelvich)		2016-03-16 07:48	2016-06-05 01:54
pg_receivexlog, pg_basebackup and data durability	Needs review	Michael Paquier (michael-kun)			2016-05-13 06:41	2016-05-13 22:07

The Commitfest App - Patch Information

Fix checkpoint skip logic on idle systems by tracking LSN progress

Edit | Comment | Status

Title	Fix checkpoint skip logic on idle systems by tracking LSN progress
Topic	Bug Fixes
Created	2016-05-18 21:59:29
Last modified	2016-05-18 21:59:36 (3 months, 3 weeks ago)
Latest email	2016-05-18 21:57:49 (3 months, 3 weeks ago)
Status	2016-09: Needs review
Authors	Michael Paquier (michael-kun)
Reviewers	David Steele (dsteele) Become reviewer
Committer	
Links	
Emails	Fix checkpoint skip logic on idle systems by tracking LSN progress First at 2016-05-18 21:57:49 by Michael Paquier <michael.paquier at gmail.com> Latest at 2016-05-18 21:57:49 by Michael Paquier <michael.paquier at gmail.com> Latest attachment (hs-checkpoints-v11.patch) at 2016-05-18 21:57:49 from Michael Paquier <michael.paquier at gmail.com> Add annotation Attach thread

History

When	Who	What
2016-08-19 01:35:27	David Steele (dsteele)	Added dsteele as reviewer
2016-05-18 21:59:36	Michael Paquier (michael-kun)	Changed authors to Michael Paquier (michael-kun)
2016-05-18 21:59:29	Michael Paquier (michael-kun)	Attached mail thread CAB7nPqQcPqxEM3S735Bd2RzApNqSNJVietAC=6kfkYv_45dKwA@mail.gmail.com
2016-05-18 21:59:29	Michael Paquier (michael-kun)	Created patch record

Patch Thread

All discussion about a patch is done on the pgsql-hackers mailing list:

Fix checkpoint skip logic on idle systems by tracking LSN progress

From:	Michael Paquier <michael(dot)paquier(at)gmail(dot)com>
To:	PostgreSQL mailing lists <pgsql-hackers(at)postgresql(dot)org>
Cc:	Andres Freund <andres(at)anarazel(dot)de>
Subject:	Fix checkpoint skip logic on idle systems by tracking LSN progress
Date:	2016-05-18 21:57:49
Message-ID:	CAB7nPqQcPqxEM3S735Bd2RzApNqSNJVietAC=6kfkYv_45dKwA@mail.gmail.com (view raw)

```
Hi all,

A couple of months back is has been reported to pgsql-bugs that WAL
segments were always switched with a low value of archive_timeout even
if a system is completely idle:
http://www.postgresql.org/message-id/20151016203031.3019.72930@wrigleys.postgresql.org
In short, a closer look at the problem has showed up that the logic in
charge of checking if a checkpoint should be skipped or not is
currently broken, because it completely ignores standby snapshots in
its calculation of the WAL activity. So a checkpoint always occurs
after checkpoint_timeout on an idle system since hot_standby has been
introduced as wal_level. This did not get better from 9.4, since
standby snapshots are logged every 15s by the background writer
process. In 9.6, since wal_level = 'archive' and 'hot_standby'
actually has the same meaning, the skip logic that worked with
wal_level = 'archive' does not do its job anymore.

One solution that has been discussed is to track the progress of WAL
activity when doing record insertion by being able to mark some
records as not updating the progress of WAL. Standby snapshot records
enter in this category, making the checkpoint skip logic more robust.

Attached is a patch implementing a solution for it, by adding in
WALInsertLock a new field that gets updated for each record to track
the LSN progress. This allows to reliably skip the generation of
standby snapshots in the bgwriter or checkpoints on an idle system.
Per discussion with Andres at PGCon, we decided that this is an
optimization, only for 9.7~ because this has been broken for a long
time. I have also changed XLogIncludeOrigin() to use a more generic
routine to set of status flags for a record being inserted:
XLogSetFlags(). This routine can use two flags:
- INCLUDE_ORIGIN to decide if the origin should be logged or not
- NO_PROGRESS to decide at insertion if a record should update the LSN
progress or not.
Andres mentioned me that we'd want to have something similar to
XLogIncludeOrigin, but while hacking I noticed that grouping both
things under the same umbrella made more sense.

I am adding that to the commit fest of September.

Regards,
--
Michael
```

Attachment: hs-checkpoints-v11.patch
Description: invalid/octet-stream (17.5 KB)
Attachment: hs-checkpoints-v11-2.patch
Description: invalid/octet-stream (1.7 KB)

Patch Assessment

Read the entire thread relating to the patch. Sometimes there will be more than one, but they should all be listed on the patch page. Then ask:

- Is there consensus on how this patch should work?
- Have the details of this patch been finalized?
- Has a patch been provided that addresses all concerns?

If the answer to all these questions is yes, then proceed to review the patch. Otherwise, try to determine what is holding the patch up:

- Is the author waiting on a consensus?
- Are there technical issues?
- Anything you can do to help?

Replying to a Thread

Reply to a thread by saving the raw message to a file (.eml) and then loading it into your email client.

- Stay on topic
- Be professional
- Don't "top post"

Read the introduction to mailing lists (`https://wiki.postgresql.org/wiki/Mailing_Lists`).

Remember that your posts are visible to the public and archived indefinitely. See the archives policy (`https://wiki.postgresql.org/wiki/Archives_Policy`) for more information.

Declare Your Intentions

- Respond on the patch thread and declare which parts of the patch you are planning to review:
  - Documentation, functional testing, performance analysis, etc.
  - Informs other potential reviewers of coverage so they can address different areas or move on to another patch.
- It is perfectly fine to have more than one person reviewing any given area. A larger/complicated patch should have more reviewers while a small/simple patch can get by with one or two.
- Remember, the committer will always review all aspects of the patch, but we would like to save them as much time as possible.

Documentation Review

PostgreSQL documentation is written in SGML. For example:

```
diff --git a/doc/src/sgml/backup.sgml b/doc/src/sgml/backup.sgml
index 7413666..9092cf8 100644
--- a/doc/src/sgml/backup.sgml
+++ b/doc/src/sgml/backup.sgml
@@ -592,7 +592,7 @@ <title>Setting Up WAL Archiving</title>

   <para>
    To enable WAL archiving, set the <xref linkend="guc-wal-level">
-   configuration parameter to <literal>archive</> or higher,
+   configuration parameter to <literal>replica</> or higher,
    <xref linkend="guc-archive-mode"> to <literal>on</>,
    and specify the shell command to use in the <xref
    linkend="guc-archive-command"> configuration parameter. In practice
```

If the feature is user-facing then the SGML documentation should explain how it works. If the feature is only exposed internally then there may not be any SGML documentation but there should still be code comments and an explanation in the email thread.

Documentation Review - Continued

- Documentation is an excellent area for non-coders to contribute. It can be difficult for patch authors to write good user-facing documentation and they appreciate another take on how to best explain a feature.
- In addition, many patch authors are not native English speakers. If you are fluent don't be afraid to fix grammatical mistakes or clean up awkward phrasing.
- Look for areas where a potential user of the feature may misunderstand the intent. Clarify where possible while keeping the documentation concise.

Apply and Compile the Patch

Compiling PostgreSQL after applying a patch is another area for non-coders to contribute.

```
git clone git://git.postgresql.org/git/postgresql.git
cd postgresql
./configure

git apply ../basebackup-exclusions-v3.patch
make install
```

Patch Does Not Apply/Compile

- It is not uncommon for a patch to get out of date during a commitfest when many other patches are being committed to master. If the patch does not apply or compile on HEAD (i.e., the most recent commit in the master branch) then try to determine the cause of the issue.
- If the patch does not apply/compile on HEAD then try to find a commit in master where it does apply/compile. Do this by picking a commit that is close to the date that the patch was originally posted, as it may be later changes in master that are causing the problem.
- Sometimes the patch author will specify a commit that is known to work in the email thread. If no commit can be found where the patch applies/compiles then reply on the patch thread with the error message you are receiving and the commits that were tested.

Test Functionality

- Functional testing requires a working knowledge of PostgreSQL but no coding experience.
- Test the patch and make sure that it does what is specified in the SGML documentation and/or the email thread. If there is a discrepancy then it may indicate an error in either the code or the documentation.
- You can either suggest a fix or simply report the issue on the email thread.

Performance Testing

- The new feature should have acceptable performance (though this is somewhat subjective).
- More importantly, the new feature should not cause a performance regression in an existing feature without good reason.
- Performance testing is often done by developers with access to unique/large scale systems.

Code Review

- Final code review is the last step and should generally be undertaken by coders experienced in the language and conversant with the project coding standards.
- However, the doesn't mean that less experienced coders can't jump in and try to figure out what is going on, especially when the patch does not apply/compile or functional testing fails. Any information you can give the authors is appreciated.
- Also, the best way to learn how to do a good review is to give it a try and see what kind of feedback you get. Over time this feedback will lead to greater confidence and ability.

Questions?

website: `http://www.crunchydata.com`

email: david@crunchydata.com

slides & demo: `https://github.com/dwsteele/conference/releases`